NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

STREAMLINING COMPLIANCE VALIDATION THROUGH AUTOMATION PROCESSES

by

Alex C. Hudson
Richard T. Leitner

March 2014

Thesis Co-Advisors: John Gibson
Karen L. Burke
Second Reader: George Dinolt

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	March 2014	Master's Thesis

4. TITLE AND SUBTITLE
STREAMLINING COMPLIANCE VALIDATION THROUGH AUTOMATION PROCESSES

5. FUNDING NUMBERS

6. AUTHOR(S) Alex C. Hudson and Richard T. Leitner

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.
IRB protocol number ___N/A_____.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited	A

13. ABSTRACT (maximum 200words)

This thesis analyzes some of the processes, tools, and content used in the certification and accreditation of Department of Defense information technology systems. The result of this analysis identifies the areas that would be improved by streamlining compliance validation through continuous monitoring and automating processes. The output of this research will be used to determine a set of requirements that, if met, would allow for the creation of a system that could be used to reduce the cost associated with compliance testing of network devices and servers, while increasing the accuracy and frequency of compliance validation. A result of this thesis will be a proof-of-concept tool that will be evaluated for functionality and used as a starting point for further discussion on future development.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Information assurance, certification and accreditation (C&A), continuous monitoring, compliance validation.	215
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

STREAMLINING COMPLIANCE VALIDATION THROUGH AUTOMATION PROCESSES

Alex C. Hudson
Civilian, Department of the Navy
B.S., Clemson University, 1999

Richard T. Leitner
Civilian, Department of the Navy
B.S., University of South Carolina, 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
March 2014

Author: Alex C. Hudson
Richard T. Leitner

Approved by: John Gibson
Thesis Co-Advisor

Karen L. Burke
Thesis Co-Advisor

George Dinolt
Second Reader

Peter Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This thesis analyzes some of the processes, tools, and content used in the certification and accreditation of Department of Defense information technology systems. The result of this analysis identifies the areas that would be improved by streamlining compliance validation through continuous monitoring and automating processes. The output of this research will be used to determine a set of requirements that, if met, would allow for the creation of a system that could be used to reduce the cost associated with compliance testing of network devices and servers, while increasing the accuracy and frequency of compliance validation. A result of this thesis will be a proof-of-concept tool that will be evaluated for functionality and used as a starting point for further discussion on future development.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ACAS	Assured Compliance Assessment Suite
AMP	Apache-MySQL-PHP
ANSI	American National Standards Institute
ARF	asset reporting format
AI	asset identification
AIS	automated information system
APS	asset publishing service
AV	antivirus
C&A	certification and accreditation
CCE	common configuration enumeration
CCSS	Common Configuration Scoring System
CEO	chief executive officer
CMRS	Continuous Monitoring and Risk Scoring
CNDSP	computer network defense service provider
CPE	Common Platform Enumeration
CVE	Common Vulnerabilities and Exposure
CVSS	Common Vulnerability Scoring System
DAA	designated approving authority
DIACAP	DoD Information Assurance C&A Process
DISA	Defense Information Systems Agency
DoD	Department of Defense
EUD	end user device
FIPS	Federal Information Processing Standard
FSO	field security operations
GIG	global information grid
GPO	group policy object
GUI	graphical user interface
HBSS	host based security system
IA	information assurance

IAVA/Bs	information assurance vulnerability alerts and bulletins
IAVM	information assurance vulnerability management
IE	Internet Explorer
IP	Internet Protocol
IS	information system
ISO	International Organization of Standards
ISS	Internet Information Services
IT	information technology
JTFTI	Joint Task Force Transformation Initiative
LAMP	Linux AMP
MAC	mission assurance category
MHS	military health systems
MS	Microsoft©
NCSD	National Cyber Security Division
NIST	National Institute of Standards and Technology
NVD	National Vulnerability Database
OAM	operation attribute model
OCIL	Open Checklist Interactive Language
OS	operating system
OVAL	Open Vulnerability and Assessment Language
POA&M	plan of action and milestones
PVS	Passive Vulnerability Scanner
RAM	random access memory
RBAC	role based access control
REM	Retina Events Manager
RDBMS	Relational Database Management System
RMF	Risk Management Framework
SCAP	Security Content Automation Protocol
SecCM	security-focused configuration management
SME	subject matter expert

SOE	standard operating environment
SQL	Structured Query Language
SRR	system readiness/review
SSH	secure shell
SSL	secure sockets layer
STIG	Security Technical Implementation Guide
TAR	tape archive
TLS	transport layer security
TMSAD	trust model for security automation data
VMS	Vulnerability Management System
XCCDF	Extensible Configuration Checklist Description Format
XML	Extensible Markup Language
XSS	cross-site scripting

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

We are grateful to SPAWAR Systems Center Atlantic for the opportunity to pursue this degree from the Naval Postgraduate School. To our advisors, Professor Karen Burke and Professor John Gibson, thank you for all your support and hard work throughout this process. Without your assistance, this would not have been possible. To our friend, Clay Stuckey, thank you for taking the time to help us with some of the technical difficulties of the proof-of-concept. Your generosity is an inspiration. To our wives, Angel Leitner and Kimberly Hudson, thank you for your patience, understanding, and support as we worked to complete this task. We love you.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Intel co-founder Gordon E. Moore described what would eventually be termed Moore's law in his 1965 paper, "Cramming More Components onto Integrated Circuits" [1]. He observed integrated circuit component density doubling every 12 months. As a result, the cost per transistor per integrated circuit decreased every year. This demonstrates that while computing power increase the costs to the consumer continue to decrease. Moore predicted that this trend would continue for at least another decade. In fact, the trend for the most part has continued to present day.

For example, in 1968 Hewlett Packard sold the 40 pound "portable" 9100A personal computer [2] for $4900, which would be over $32,000 [3] in today's money. It was billed as a personal computer capable of scientific and engineering computations utilizing up to 16 data storage registers. By contrast, the Raspberry Pi is a credit card sized computer that is sold for $35 and comes "stock" with 512MB of random access memory (RAM) and an ARM11 processor capable of 700 million operations per second while weighing in at 1.6 ounces [4]. As a result of these trends and ubiquitous network connectivity, we find more and more computers being used in the government, private sector and our homes. For example the number of personal computers in use worldwide reached one billion in 2008 and by the year 2014 there are estimated to be over two billion in use [5].

A. PROBLEM SCOPE

The United States government's reliance on computing technologies and its connectivity to public networking

infrastructure positioned it on a warfare domain with an ever expanding battlefront in which any adversary with a computer can engage in battle. According to a July 2011, report generated by the U.S. Government Accountability Office regarding cyber efforts,

> The U.S. military is dominant in the land domain, unchallenged in the air, and has few near-peers in the maritime domain. However, the technical and economic barriers to entry into the cyber domain are much lower for adversaries and as a result place U.S. networks at great risk. [6]

The rapid growth of information technology (IT) systems and reliance on technology present unique challenges for the Department of Defense (DoD) concerning IT Security. The integration of new technologies and systems into the everyday work-life of DoD employees has introduced a reliance on these systems in order to function. As new systems are introduced and existing systems upgraded to provide additional security or function more potential vulnerabilities are introduced, a result of the growing complexity of systems. According to Symantec, in 2011 there were 4,989 new vulnerabilities reported, which works out to be approximately 95 new vulnerabilities reported per week [7]. Both the growing number of vulnerabilities being introduced daily and the trend of system component growth are increasing the time and resources required to secure systems.

B. THESIS SCOPE

The primary focus of this thesis is to examine the effects that the growing number of computing devices, as well as the ever increasing levels of computing power, has

on the process for securing an environment within the DoD. Relevant information assurance (IA) processes, standards, and tools are discussed and analyzed with an emphasis on supporting continuous monitoring and automated validation. The output of this research is a list of requirements for constructing a toolset to monitor and assess IT devices and a proof-of-concept tool to demonstrate the requirements.

C. ORGANIZATION OF THESIS

The main content is divided into four additional chapters following the introduction. First, the current certification and accreditation (C&A) IA processes and tools for validating assets and maintaining compliance are evaluated in Chapter II. Additionally, the difficulties associated with maintaining a secure environment as these assets grow in number and interconnectivity is also discussed. Chapter III proposes a set of requirements for meeting these challenges and discusses possible options for satisfying them. Chapter IV details a proof-of-concept system built to satisfy the requirements posed in Chapter III, while Chapter V details how the system was validated for functionality. Finally, Chapter VI evaluates whether the proof-of-concept system is viable, the effect it could have on compliance monitoring and validation, and what improvements or further development should take place.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

A. INTRODUCTION

DoD funded organizations are tasked with evaluating the security posture of networking devices and servers against the security technical implementation guides (STIGs) provided by Defense Information Systems Agency (DISA) as part of a site or type accreditation. The current process for these evaluations typically involves a C&A team funded for the purpose of executing security audits on each applicable system component and providing vulnerability assessment reports to the system owners. This team must interface directly with system owners to coordinate scans on each device, often requiring hands-on assistance. This process is repeated prior to any scheduled accreditation event or during routine evaluations against the system's accredited baseline.

The current process calls for fully funded engineers with intimate working knowledge of each system component to work alongside the C&A team during the evaluation period. Unfortunately, it is unrealistic from a technical or financial perspective to hire engineers dedicated to supporting these tasks.

Typically, during the evaluation period project funded engineers are pulled from current tasking, which interrupts their project workflow, in order to complete these C&A tasks. It is inefficient to rely on project funded engineers to complete these tasks as it often results in a loss of momentum in their primary project tasking in addition to a potential conflict of interest. It is often

during these evaluation periods that these systems are discovered to be out of compliance, which requires the C&A evaluators revalidate once the system has been brought back into compliance, further impacting the collaterally tasked engineers.

Several commercially available enterprise tools exist that meet some of these needs. There are tools, for example Retina and Nessus, which provide an automated way of evaluating a component's security baseline. Unfortunately, these types of tools are geared mostly towards information assurance vulnerability management (IAVM) compliance and are not ideal tools to provide continuous system monitoring. Other commercial tools from companies like EiQ Networks and Refense Technologies provide a means of continuously monitoring the target environment and an opportunity to react in real-time to non-compliance issues, but are costly.

From a DoD perspective, DISA has been providing STIG guidance in the form of checklists with limited system readiness/review (SRR) scripts and Security Content Automation Protocol (SCAP) content. The DISA Gold Disk had been the primary automated tool for evaluating STIG compliance on supported platforms. It primarily supported "the ability to detect installed products, identify and remediate applicable vulnerabilities and generate a file that can be used for asset registration and findings upload into DISA's vulnerability management system (VMS)" [8]. However, as of late 2012, DISA stopped providing updates for the DISA Gold disk utility and has focused primarily on supporting the SCAP standard.

DISA is continuing development of a Continuous Monitoring and Risk Scoring (CMRS) system that takes a risk management approach to providing a quantitative view of an organizations security posture. At this time there is no widely adopted automation or continuous monitoring integrated into the network and system compliance validation process, which leads to an extensive amount of resources being dedicated to these tasks. For example, the manual process to validate STIG compliance against network devices can take hours per device and even then the likelihood of error or omission is high because the reviewer is often the same person who configured the device.

There would be great value in an open source system or tool set that utilizes a standard framework for evaluating system security baselines. Such a tool should take as input custom templates based on a standard framework that would allow users to share, create and customize security compliance templates to meet their specific organizational needs. Providing an open source tool to the DoD community would allow organizations to adopt its use and would encourage further development of custom templates and refinement of existing templates to be used by the community as a whole.

B. CURRENT PROBLEMS FACING DOD IT SECURITY

The rapid growth of IT systems and technology present unique challenges for the DoD concerning IT security. Consider that:

> For the top brass, computer technology is both a blessing and a curse. Bombs are guided by GPS satellites; drones are piloted remotely from across the world; fighter planes and warships are now huge data-processing centres; even the ordinary foot-soldier is being wired up. Yet growing connectivity over an insecure internet multiplies the avenues for e-attack; and growing dependence on computers increases the harm they can cause. [9]

The integration of new technologies and systems into the everyday work-life of DoD employees has introduced a reliance on these systems in order to function. As new systems are introduced and existing systems upgraded to provide additional security or functionality, more potential vulnerabilities are introduced as these systems become more complex. The Common Vulnerabilities and Exposure (CVE) dictionary developed in 1999 by the Mitre Corporation and currently funded by the Office of Cyber Security and Communications, provides a common naming convention for listing information security vulnerabilities and exposures for openly published software security flaws. The Mitre Corporation defines vulnerability as a mistake in software that can be leveraged by an attacker to gain unauthorized access to a system or network, while an exposure is defined as mistake in software provides access to information of capabilities that could be used by an attacker as a vehicle to gain access to a system or network. Figure 1 shows the number of CVEs reported by year from the National Institute of Standards and Technology (NIST) between 1988 and 2013 according to the CVE Statistics Query Page for the National Vulnerability Database (NVD) [10].

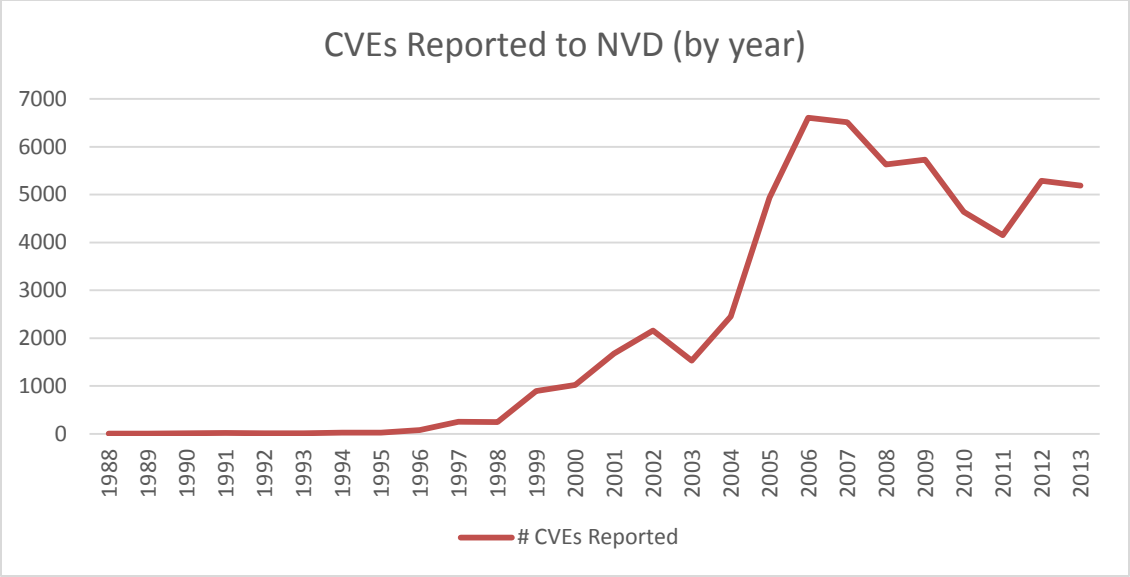


Figure 1. Number of CVEs reported per year

The increase in vulnerabilities introduced each year, as depicted in Figure 1, can be attributed to at least two things: new applications being introduced to market and products becoming more complex as they introduce additional features and capabilities. These changes in number and complexity alter the vulnerability landscape and introduce new avenues for exploitation. Figure 2 shows the top three CVE vulnerability categories reported by year.

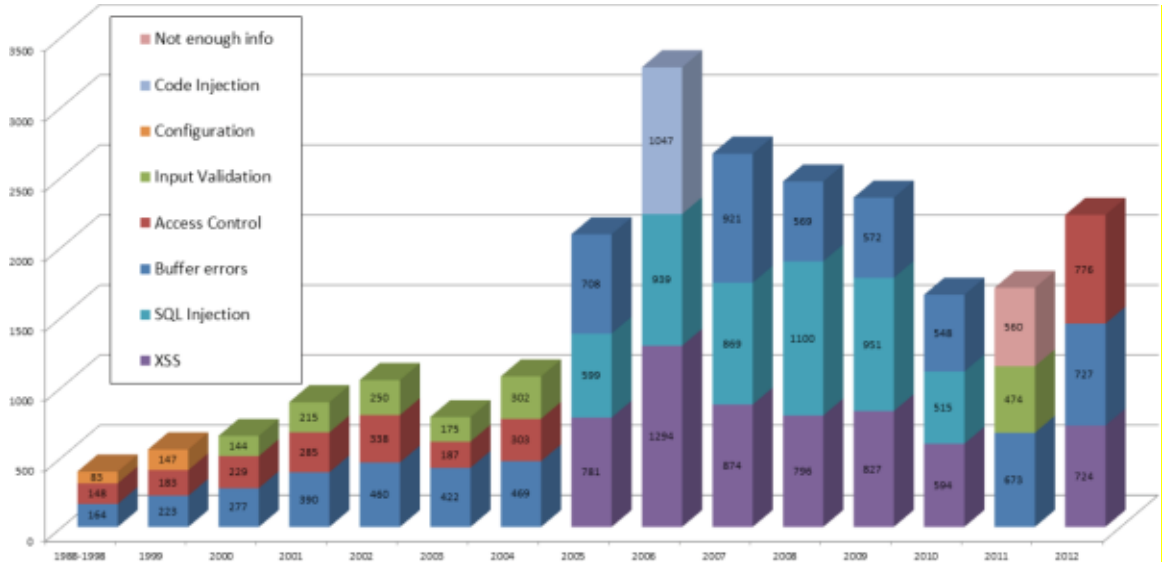


Figure 2. Top three CVE Categories (by year)

The introduction of new types of vulnerabilities may attribute to the spikes in reported vulnerabilities. The declines in reported vulnerabilities may be the result of product vendors patching existing software and learning to develop future software with additional safeguards and protections. For example, in 2005 cross-site scripting (XSS) and Structured Query Language (SQL) injection vulnerabilities show up in the top three with 2006 seeing the introduction of code injection exploits as well [11].

NIST explains these vulnerabilities, documented as CVEs, are categorized and maintained within the NVD that is

> a comprehensive database of cyber security vulnerabilities in IT products that was developed by NIST with the support of the National Cyber Security Division (NCSD) of U.S. Department of Homeland Security. [12]

The growing number of vulnerabilities being added daily to the NVD provides a staggeringly large avenue for exploitation considering the DoD currently operates more

than 15,000 different computer networks across 4,000 military installations around the world. On any given day, there are as many as seven million DoD computers and telecommunications tools in use in 88 countries using thousands of warfighting and support applications. [13]

Given the increasing exposure to exploitation, due to the growing number of software vulnerabilities and attack vectors, the cyber domain has become as relevant as the traditional domains of land, sea, air, and space.

While computing power is getting faster and cheaper for consumers and industry, these resources are also becoming more readily available for conducting cyber warfare. According to a July 2011 report on DoD cyber efforts:

> The U.S. military is dominant in the land domain, unchallenged in the air, and has few near-peers in the maritime domain. However, the technical and economic barriers to entry into the cyber domain are much lower for adversaries and as a result place U.S. networks at great risk. [6]

On the cyber front the US is fighting a war where all one needs is a computer with an internet connection to compete. The February 2010 *Quadrennial Defense Review* has this to say:

> It is therefore not surprising that DoD's information networks have become targets for adversaries who seek to blunt U.S. military operations. Indeed, these networks are infiltrated daily by a myriad of sources, ranging from small groups of individuals to some of the largest countries in the world. [13]

As technology and interconnectivity become more integrated into the other traditional domains the

importance of protecting and establishing a dominant presence in the cyber domain is greatly increased. One tactic employed by the government to foster this dominance is through the use of C&A.

C. C&A PROCESS AND PURPOSE

1. Overview

C&A is a federally mandated, formal process for identifying, implementing, and managing IA requirements, controls and services with an emphasis on maintaining them throughout the system lifecycle. To deconstruct the terminology, the National Computer Security Center states that certification is:

> the comprehensive assessment of the technical and nontechnical security features and other safeguards of a system to establish the extent to which a particular system meets a set of specified security requirements for its use and environment, [14]

while, accreditation is:

> the formal declaration by the Designated Approving Authority (DAA) that an automated information system (AIS) is approved to operate in a particular security mode using a prescribed set of safeguards and should be strongly based on the residual risks identified during certification. [14]

2. DIACAP

The Department of Defense Information Assurance Certification and Accreditation Process (DIACAP) is the DoD's official process for C&A. DIACAP can be broken into five distinct activities, as shown in the following process wheel diagram in Figure 3 [15].

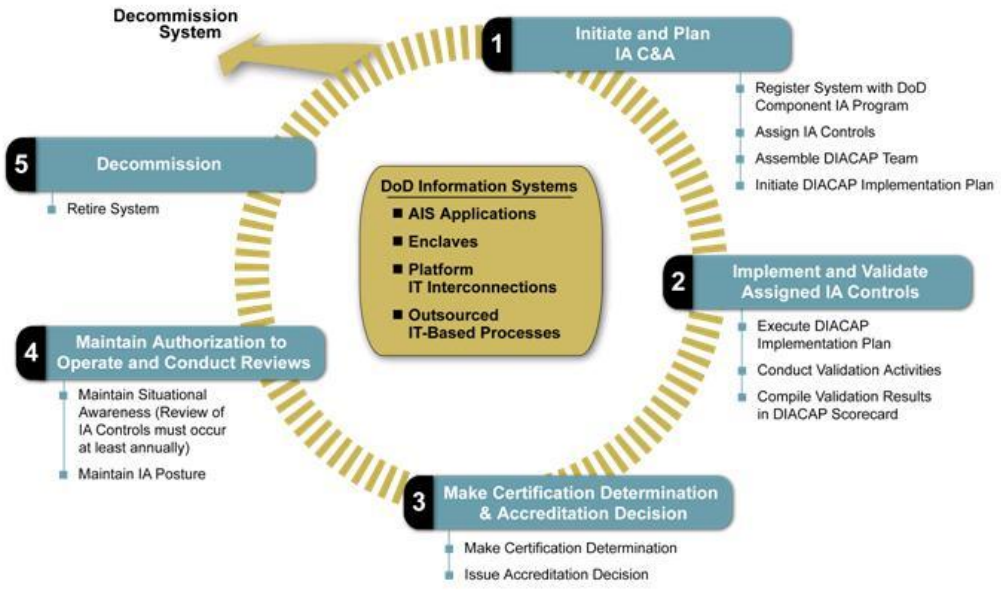


Figure 3. The Five DIACAP Activities

Initiating and planning IA C&A is listed as the first activity. This is where the DIACAP team is assembled and the system is registered with a DoD component IA program. It is also when IA controls are assigned and concurrence for the implementation plan is determined.

Implementation and validation of assigned IA controls is the next activity and it is here where the greatest impact of automated validation tools can be made. After the DIACAP implementation plan is executed, validation activities are conducted and validation results are compiled into a DIACAP scorecard. Today, certain automated tools, such as vulnerability scanners, SRRs and the DISA Gold disk, can be used to conduct portions of the validation activities. Commercial software exists that allow for network device evaluation to be automated as well. The use of automated tools should increase efficiency

and accuracy through the minimization of human error. The resulting artifact of the validation activities is a scorecard that is used during the next step.

The third activity is to make the certification determination and the accreditation decision. In short, the risks, vulnerabilities, mitigation costs, and exposure are all weighed and a recommendation is made. This recommendation, the business and mission needs, along with the likelihood and potential impact of any loss of confidentiality, integrity or availability suffered by the system would then be weighed by the accrediting body and a decision made to accredit or not accredit the system. If accredited, the system would enter the fourth activity of DIACAP.

In the fourth activity, the authorization to operate is maintained and annual reviews are conducted. This is another area where automated validation tools can have a significant impact. In the second activity, the tools were used to evaluate a system from scratch. In this activity the tools can be used to continuously monitor a system to insure it remains in compliance. Such tools can also be used during any re-accreditation, typically due to system upgrade or modification, since they will be able to provide an up-to-date validation compliance report.

The final activity associated with the DIACAP process is decommissioning. This activity is initiated when the decision is made to retire a system. In order to retire the system the DIACAP registration information, system related data and supporting IA objects or core services in the DoD's global information grid (GIG) must be disposed.

3. Risk Management Framework

The traditional C&A process has been transformed into a common framework whose goal is to "improve information security, strengthen risk management processes, and encourage reciprocity among federal agencies" [16]. NIST publication 800-37, developed by the Joint Task Force Transformation Initiative (JTFTI) Working Group, created a six-step process for risk management called the Risk Management Framework (RMF). The main tenants of the RMF include: (i) "baking in" of information security capabilities through the use of management, operational and technical security controls; (ii) continuous awareness of information system (IS) security through monitoring processes; and (iii) the delivery of needed information to senior leaders in an efficient manner that allows them to make decisions relative to risk management.

The overall RMF process is illustrated in Figure 4 [16].

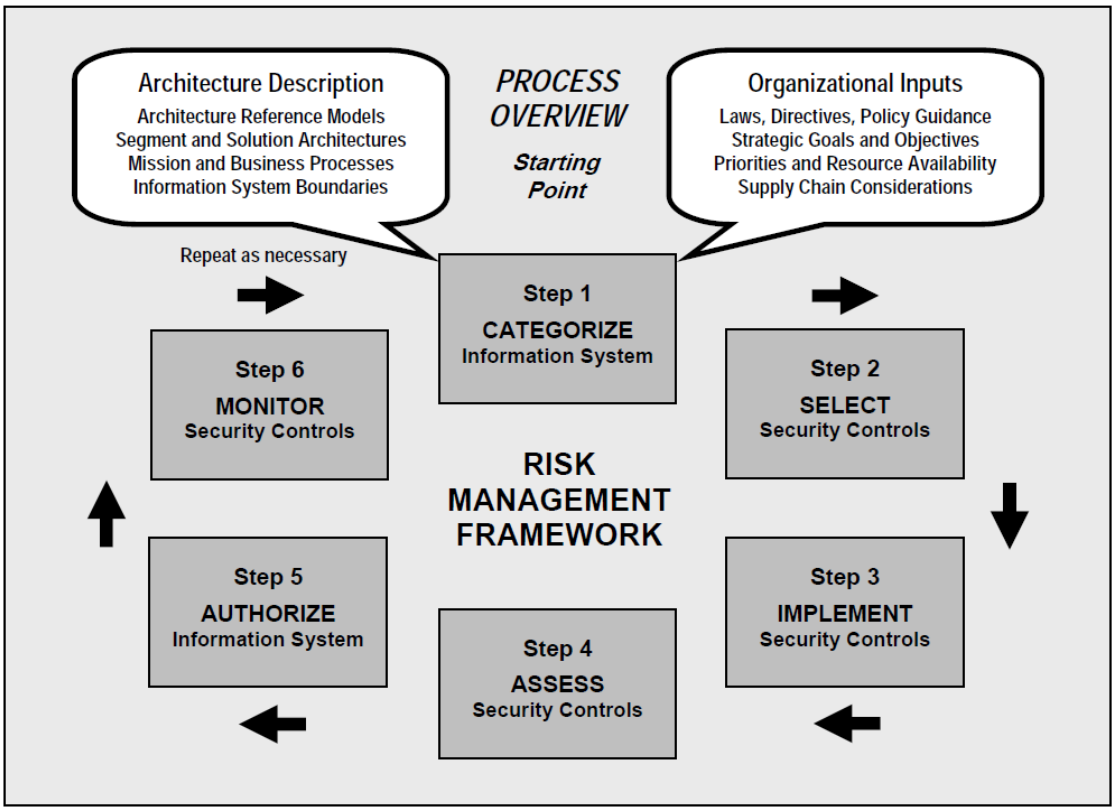


Figure 4. Risk Management Framework

The first step is to categorize the system. This requires understanding how the information will be used, how it will be transmitted, and how it will be stored. It also requires understanding the impacts associated if that information's confidentiality, integrity, or availability is compromised.

Once the system is categorized, security controls can be selected. Initially, a baseline set of controls is assigned but as risk is assessed and local conditions are taken into account the set of selected controls may be supplemented or tailored to meet specific needs.

The third step centers on implementation of the selected security controls. It is also during this step

that time is taken to explain how the controls are implemented within the information system and its operating environment.

The fourth step is where the implemented security controls are assessed. Someone trained in the appropriate assessment protocol, called a validator, is looking to ensure that the selected security controls have been implemented properly and are working correctly and are achieving the desired results. Due to the nature of the work in this step, it is expected that a validation automation tool would or could have significant positive impact on both the results and efficiency of this activity.

Once the assessment is complete, a decision is made based on the results of the assessment and the determination of risk associated with operation of the information system. If the risk is acceptable to the organization in charge of the decision, then the system is authorized for use. If not, additional work must be done to get the system security posture suitable for authorization.

Once a system is authorized for use, monitoring of the system begins. In this step, the security controls are assessed in the same manner as they were during step four including assessing the effectiveness of the controls and documenting any changes to the system or the operational environment. It is also during this step that any changes made to the system are analyzed for risk impact and additional risk acceptance decisions from organizational officials be obtained as required. Obviously, an automated validation and continuous monitoring solution would allow

the organization to track changes while maintaining a constant picture of the information system's security posture.

4. Lasting Effects

All too often security is an afterthought during the various phases of the system life cycle. Fortunately, no matter the phase, initiation, development and acquisition, implementation, operations and maintenance, or disposal and retirement, the C&A process can still be applied to great effect. Whether DIACAP or RMF is chosen, C&A is a powerful process that if utilized properly, can manage the security of a system throughout its life cycle. A system that allows for more rapid and consistent validation and monitoring of security controls also allows C&A processes to better fulfill their purpose.

D. CONTINUOUS MONITORING AND COMPLIANCE VALIDATION TOOLS

> Millions of dollars and thousands of hours are spent on C&A, and C&A levels are used to assess security. In reality C&A is a 20-year-old paperwork exercise that does not yield improved security. The only real way to measure security is to track the numbers and types of compromise over time, and try to see that number decrease.
>
> Richard Bejtlich, President & Chief Executive Officer (CEO) of TaoSecurity [17]

While Mr. Bejtlich may be exaggerating the ineffectiveness of C&A, his statement does highlight two issues with the current C&A process: the cost and time associated with the effort and the real world implication that the true measure of security for any given system will be seen over an extended period of time. While capturing

these costs can be difficult, tools that can automate any portion of compliance validation could have significant impact on both the cost and time associated with these events. Tools that can provide a means to continuously monitor systems would help counter the "set it and forget it" mentality that implies the C&A process is largely a paper drill with no lasting effect on the system security.

In order to provide sufficient support during C&A events, management must plan to have privileged subject matter experts (SME) available to support the validator's specific system component reviews. The process for completing an evaluation of a system component is cumbersome and requires an exhaustive review of the system component against the last DISA provided STIG.

The DISA field security operations (FSO) provide technical guidance for locking down IA systems and software through STIGs. In addition to STIGs, the DISA FSO also provides STIG checklists, which are detailed instructions for performing configuration validation and remediation against applicable STIGs for an IA asset. DISA publishes all current versions of STIGs and STIG checklists to https://iase.disa.mil/stigs. DISA also publishes SRR Scripts, which are custom built tools for performing automated STIG compliance validation. The most frequently used SRR tool is the DISA Gold disk that provided STIG validation against the most current Microsoft Windows operating systems. As of December 2012, support for the DISA Gold disk terminated and current efforts are focused on providing SCAP content for new/updated DISA STIGs.

Typically, the STIG for a system component is available in a generic or device specific checklist or system readiness/review (SRR) scripted application. While the availability of the checklists and SRRs provide significant time savings and structured guidance during the evaluation process they are limited in scope. Many devices do not have a device specific checklist; this then requires a degree of interpretation by the C&A team when evaluating a system component against a generic device STIG. While SRR scripted applications are available for most MS Windows and Linux/Unix based operating systems (OS) and most common software suites, they are virtually non-existent for network devices thereby requiring a manual review for each component.

For example, in the past the STIG review process for MS Windows-based servers often involved running the latest version of the DISA Gold disk for the Windows OS and many major Windows applications (e.g., Internet Explorer (IE), Microsoft Office, and Antivirus (AV)). The DISA Gold disk from July 2012 was used to evaluate a generic Windows 2003 Member Server (e.g., not a domain controller or DNS/DHCP server). Table 1 was constructed using these scan results to show the components reviewed, the number of automated checks, the number of manual checks, and a percentage of the total number of checks that are automated.

Gold Disk Automation Percentages (Windows 2003 R2)				
Component	Automated	Manual	Total	%Automated
.NET Framework 1.1	1	45	46	2%
Framework 3.5	1	45	46	2%
Antispyware	0	17	17	0%
McAfee Antivirus	0	72	72	0%
Desktop Apps	0	5	5	0%
Office - Word 2003	5	2	7	71%
Windows 2003	474	212	686	69%
IE 7	104	3	107	97%

Table 1. Gold Disk Automated Checks

The absence of automation within the SRR utility adds labor hours and additional cost to each system component reviewed. For example, the Application Virtualization Hosting Environment under DoD Military Health Systems (MHS) manages 1500 Servers for hosting applications for MHS users.

The DISA-provided SRRs and SCAP content provide for some measure of automation regarding servers and end user devices (EUDs) such as desktop and laptop computers, but at this time the checklists they provide for networking devices are primarily used as a guide to complete manual validation checks. In many ways this is to be expected. In the case of servers and EUDs, the OS and installed applications are leveraged to run the automation scripts and create the compliance reports. Networking devices are often by design special purpose and usually run code specifically designed to support the device's primary function. While these devices might offer standard methods of access and configuration backup, the wide range of

proprietary software supporting these products makes it difficult to create any standard tools that run on the devices themselves.

Networking devices comprise the foundational infrastructure that makes server and EUD communication possible. Besides supporting all communication between servers and EUDs and providing these devices connections to larger networks, networking devices often serve as the first line of defense from unauthorized access to computing networks. When comparing sheer numbers, networking devices make up a very small portion of those devices connected to the internet. The role of network devices in supporting network connectivity and defense places them at points in the architecture that increase their exposure to potential enemies. They are both the first line of defense and the most easily visible from the Internet. Additionally, their various roles in the architecture also make them high impact targets. In many cases, the exploitation of a single network device can result in loss of confidentiality, integrity and availability of mission essential resources. This makes network device security compliance of paramount importance.

As mentioned previously compliance validation of network devices is a manual process. According to *Military Information Technology* magazine's article, "Automatic for Security":

> That manual process can take between 45 minutes and 2 hours per device, and it must be done by a very skilled engineer with networking credentials and certifications to confirm the device

> configuration. Not only is this labor intensive, but it is also difficult to achieve a high degree of accuracy. [18]

A tool that could automate this process would go a long way toward ensuring that network security settings were being implemented in a standard and accurate way across the DoD. Additionally, if this tool had a means of continuously monitoring these settings across the enterprise, then security configurations could be more consistently maintained over longer periods of time therefore reducing the number of vulnerabilities exposed to the enemy. Of course, a common standard for DoD security personnel to write and share compliance validation content would prevent duplicate work and aid in implementation of standardized checks. To meet this goal, NIST created a framework for using specific standards-enabled automated compliance validation.

E. SECURITY CONTENT AUTOMATION PROTOCOL

SCAP is a standardized set of specifications that compose a framework, designed to promote the automation of security compliance validation and detection while maintaining interoperability across a wide range of security products that vary in function and scope. SCAP is composed of 11 components in five categories, which are listed in Table 2, as part of the SCAP 1.2 specification [19].

SCAP Component	Description
Languages	
Extensible Configuration Checklist Description Format (XCCDF) 1.2	A language for authoring security checklists/benchmarks and for reporting results of evaluating them
Open Vulnerability and Assessment Language (OVAL) 5.10	A language for representing system configuration information, assessing machine state, and reporting assessment results
Open Checklist Interactive Language (OCIL) 2.0	A language for representing assessment content that collects information from people or from existing data stores made by other data collection efforts
Reporting Formats	
Asset Reporting Format (ARF) 1.1	A format for expressing the exchange of information about assets and the relationships between assets and reports
Asset Identification 1.1	A format for uniquely identifying assets based on known identifiers and/or known information about the assets
Enumerations	
Common Platform Enumeration (CPE) 2.3	A nomenclature and dictionary of hardware, operating systems, and applications, plus an applicability language for constructing complex logical groupings of CPE names
Common Configuration Enumeration (CCE) 5	A nomenclature and dictionary of software security configurations
Common Vulnerabilities and Exposures (CVE)	A nomenclature and dictionary of security-related software flaws
Measurement and Scoring Systems	
Common Vulnerability Scoring System (CVSS) 2.0	A system for measuring the relative severity of software flaw vulnerabilities
Common Configuration Scoring System (CCSS) 1.0	A system for measuring the relative severity of system security configuration issues
Integrity Protection	
Trust Model for Security Automation Data (TMSAD) 1.0	A specification for using digital signatures in a common trust model applied to other security automation specifications

Table 2. SCAP 1.2 Components

1. SCAP Languages

SCAP languages provide a vocabulary specifically designed for expressing security policy, checks, and assessments. The Open Vulnerability and Assessment Language (OVAL) is used to provide a standardized method for expressing machine readable rules to assess current system setting states defined in these rules. It provides a means for writing automated checks that can be evaluated against an asset through SCAP compliant tools. The OVAL process is shown in Figure 5 [20].

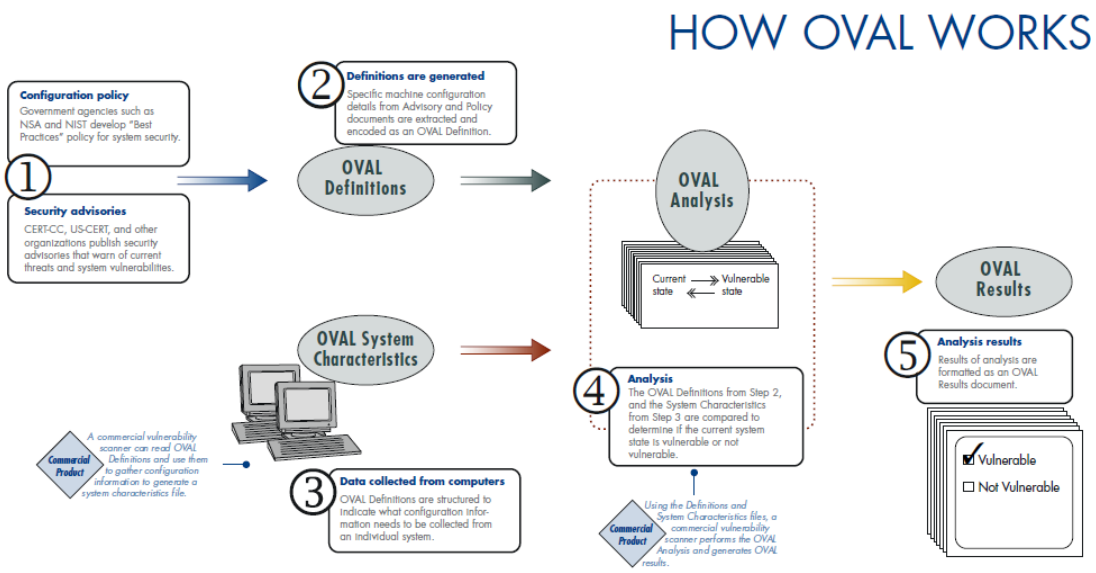


Figure 5. OVAL Overview

Typically, OVAL rules are used to evaluate a system's security configuration or software patch compliance; however, rules can be created to validate non-security machine readable settings as well. For example, content written using OVAL can be used to validate that Internet Explorer's zone configurations are set according to DISA STIG guidance as well as ensuring that the browser's homepage is set to a company's intranet site. The Open Checklist Interactive Language (OCIL) is an XML-based language that is utilized to provide a method for presenting questionnaires to users for the purpose of gathering information that is not machine-readable or harvest data from previous assessments. This enables the integration of manual checks, which currently cannot be automated, into SCAP content. OCIL can also be used to aggregate results from varied data sources and display them in a single standardized format [21].

The Extensible Configuration Checklist Description Formation (XCCDF) specification is a vender-neutral, standardized approach to documenting security checklists for automated and manual validation checks. XCCDF is written in XML that can be embedded inside existing documentation. As an example, the DISA STIG Checklists, now embedded with XCCDF content, can be read by an XCCDF tool while maintaining the same look and feel as previous versions. XCCDF also supports the integration of future content, data formats, and features without hindering the functionality of existing XCCDF tools. XCCDF does not specify how the checks are executed but instead references the OVAL and OCIL definition files that contain this information [22].

2. SCAP Enumerations

SCAP enumerations define a standardized naming convention and a list of items expressed with this standard. Common Configuration Enumerations (CCEs) are unique identifiers assigned to configuration guidance statements. Similar to CCEs, the CVEs are unique identifiers assigned to known system vulnerabilities. Common Platform Enumeration (CPE) provides the naming conventions used to identify and describe the applications, operating systems, and hardware devices being evaluated [23].

Measurement and scoring SCAP components are used to categorically examine security weaknesses and provide a quantitative measurement for each vulnerability. The Common Vulnerability Scoring System (CVSS) is a standard framework for quantifying risk of vulnerabilities introduced by

software flaws as they pertain to an organizations operating environment. CVSS is composed of three Metrics Groups, categorically grouping the metrics defined, as seen in Figure 6 [24].

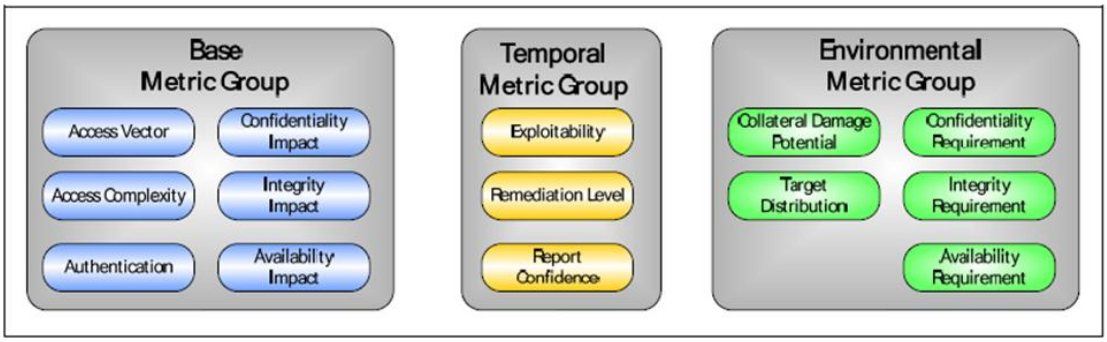


Figure 6. CVSS Metric Groups

The base metric group comprise metrics that are consistent across all environments and do not change over time. Temporal metrics represent threats to vulnerabilities that may change over time. Environmental Metrics address threats to vulnerabilities that are associated with the user's operating environment. Each group produces a score between 0.0 and 10.0 that, when used in conjunction with Federal Information Processing Standards (FIPS) 199 categories, can be used to produce impact scores tailored to the organization's operating environment. Impact scoring is used to quantify the severity of a successful exploitation for a given vulnerability as it pertains to the confidentiality, integrity, and availability of the system being evaluated.

The Common Configuration Scoring System (CCSS) is derived from CVSS and is used to quantify the severity of security configuration issue vulnerabilities. CCSS uses the

same scoring range as CVSS and is composed of the same three metric groups, with variations to the metrics within the Temporal and Environmental Metric Groups. CVSS and CCSS scoring components, integrated with SCAP content, provide the objective scoring required to quantify the risk associated with individual checks [24].

3. SCAP Reporting Formats

Reporting formats in SCAP are used to collect asset information and define how the output will be displayed. The Asset Identification framework in SCAP defines a process for using known attributes or identifiable data generated by the asset. The Asset Reporting Format (ARF) standardizes the way reports are generated and processed. The ARF can also correlate data from various sources as it pertains to a unique device that has identifiable attributes discovered through Asset Identification (AI). These reporting formats provide a vendor neutral process for identifying assets and presenting information that pertain to each asset [25].

4. SCAP Integrity Component

The SCAP integrity component, the trust model for security automation data (TMSAD), was created to provided integrity, authentication, and traceability for security automation data. The TMSAD defines a data component that can be integrated into Extensible Markup Language (XML) documents using existing standards to provide a means of generating hashes and signatures for automation data [26].

F. ASSURED COMPLIANCE ASSESSMENT SUITE

The Assured Compliance Assessment Suite (ACAS) is a software suite that provides vulnerability scanning, configuration assessment, and network discovery. ACAS was developed by DISA with collaboration from industry partners to replace the DoD's current vulnerability scanning toolset, Retina and Retina Events Manager (REM). The ACAS suite is composed of five components.

1. SecurityCenter

The SecurityCenter is a management console that provides a graphical user interface (GUI) to centrally manage assets within an organization's infrastructure that are being monitored by the ACAS scanning component. SecurityCenter also enables distributed and load-balanced scanning and customized reports for analyzing aggregate scan data [27].

2. Nessus Vulnerability Scanner

The Nessus Vulnerability Scanner enables the discovery of assets, vulnerability scanning, configuration auditing, and compliance validation.

3. Passive Vulnerability Scanner

The Passive Vulnerability Scanner (PVS) monitors real-time network traffic, using packet captures to determine the network topology and detect server and client side vulnerabilities. It is continuously monitoring network traffic, detecting new hosts, applications, and vulnerabilities and reporting this information to SecurityCenter in real-time. Figure 7 shows the Nessus and

PVS components working together as a continuous network monitoring solution [28].

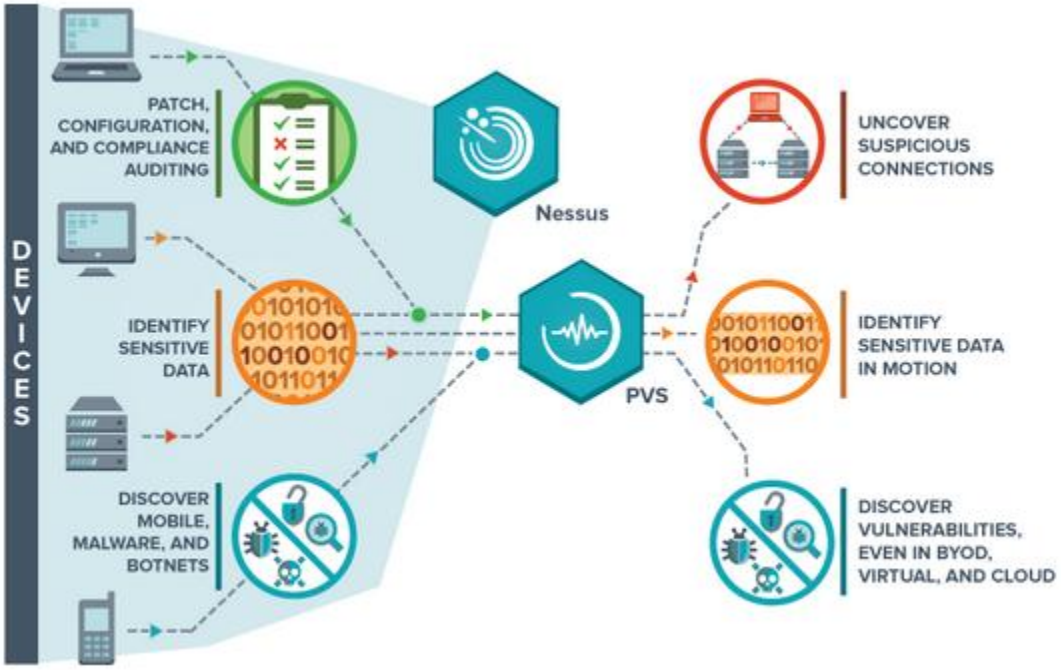


Figure 7. Nessus and PVS Data Flow

4. X-Tool

The X-Tool is a standalone tool used to convert XCCDF/OVAL files into an XML Schema that can be imported into SecurityCenter. This tool is only used for converting SCAP content into a format that can be used by SecurityCenter.

5. Topology Viewer

The Topology Viewer is used to graphically display the network map with protocols and vulnerability information created from data gathered by the PVS hosts and reported to SecurityCenter.

G. VULNERABILITY MANAGEMENT SYSTEM

DISA built the VMS to provide command and security channels within DoD a view into the current compliance state of a DoD device and the organization responsible for that asset. The C&A process utilizes VMS to record and track assets, vulnerability compliance, and manage plan of action and milestones (POA&M) for accreditation activities. VMS is also utilized to provide vulnerability notifications and track the receipt and remediation or mitigation of vulnerabilities.

The introduction of VMS provided a much-needed centralized distribution for IAVM; however, the tracking system relies on manual input for assets and tracking compliance for each asset. The manual entry aspect of VMS is very labor intensive, subject to human error, and easily manipulated. The inherent flaw of VMS is the requirement that system owners manually enter their assets, software baseline, and provide monthly scan reports. Those who choose not to utilize VMS or neglect to accurately represent the software baseline of an asset would operate undetected and potentially in a non-compliant state. Few measures are in place to dissuade "check box compliance" where an asset could be marked compliant without external validation.

The diagram in Figure 8 shows data captured from seven sites that have been transitioned by their Computer Network Defense Service Provider (CNDSP) from VMS to CMRS.

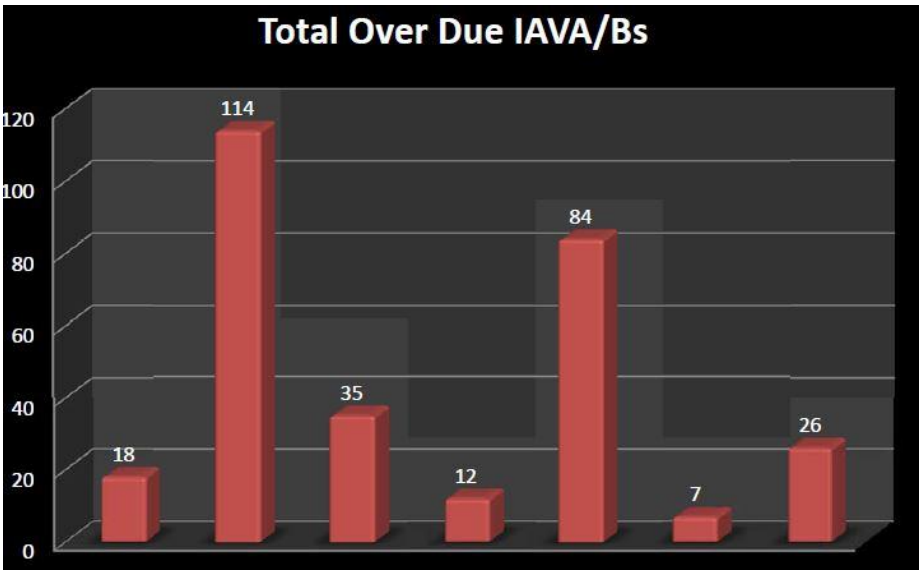


Figure 8. CMRS Report for IAVA/Bs out of compliance

Each of these sites had reported in VMS full compliance for these information assurance vulnerability alerts and bulletins (IAVA/Bs) with no outstanding POA&Ms.

H. CONTINUOUS MONITORING AND RISK SCORING

The DISA CMRS user's guide states:

> The objective of CMRS is to assess and measure the risk state of the DoD Enterprise security controls such as software inventory, security technical implementation guide (STIG) compliance, vulnerability and patch compliance, and anti-virus configurations. [29]

CMRS is a web-based security risk reporting system for DoD assets that supports the RMF and collects compliance data from automated feeds provided by host based security system (HBSS) or ACAS managed assets. Figure 9 shows the interaction between HBSS and ACAS assets reporting into CMRS [29].

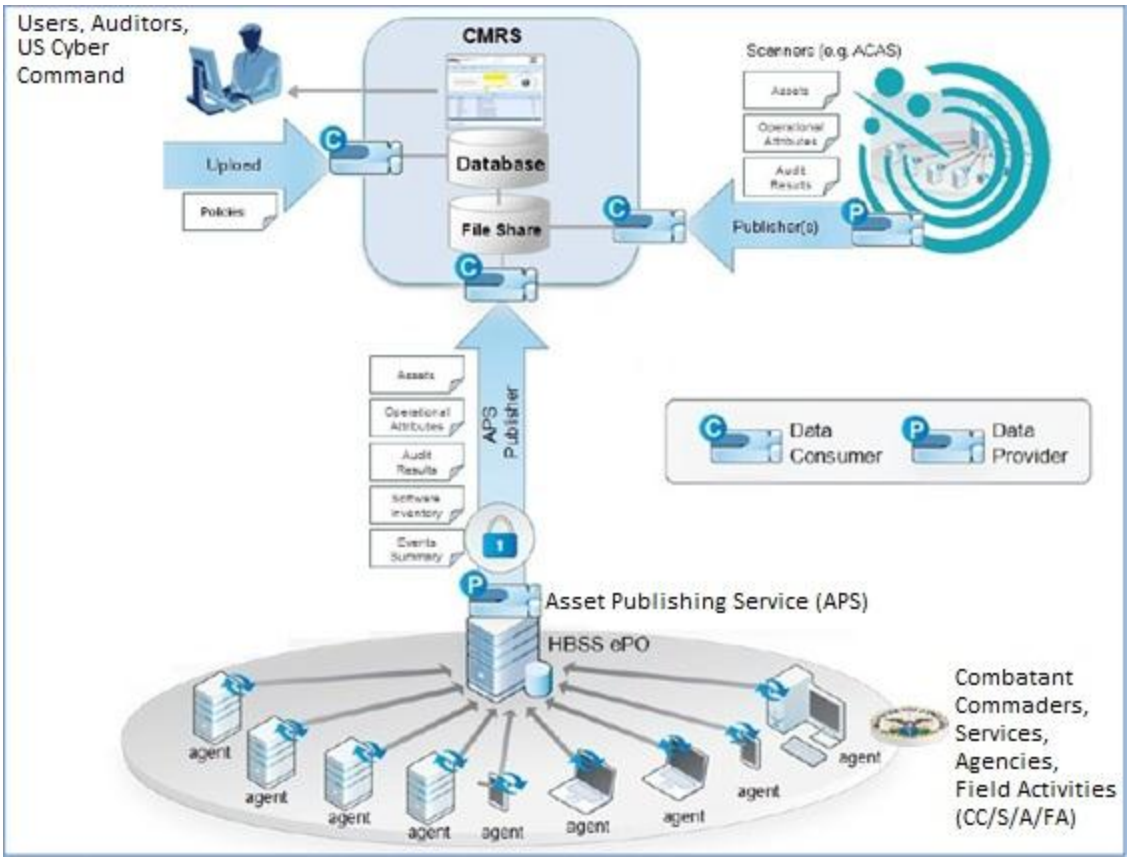


Figure 9. CMRS Data Flow

1. CMRS HBSS Asset Reporting

The HBSS solution deployed to servers, laptops, and desktops within DoD is the McAfee Endpoint Product security applications. Under CMRS HBSS functionality is extended through additional modules and capability. The Asset Publishing Service (APS) provides HBSS data (asset, audit, software inventory, and event summary) to be accessible and consumed by CMRS. The operational attribute module (OAM) allows tagging assets with operational attributes to be sent to CMRS to provide additional detail about a monitored asset.

HBSS assets are given a score from 0 to 16,000 (zero meaning no calculated risk and 16,000 being the maximum

calculated risk). CMRS calculates a risk score for each of the four risk factors (AV, Standard Operating Environment (SOE), IAVM, and STIG) with a score from 0 to 4,000. HBSS is currently the main source for CMRS asset compliance data; however, data feeds from DISA's ACAS are also supported.

2. CMRS ACAS Asset Reporting

ACAS asset reporting to CMRS is available for devices that do not support the installation of HBSS software. In addition, ACAS can provide an external look at an asset's compliance from the network side.

ACAS assets are given a score from 0 to 8,000 (zero meaning no calculated risk and 8,000 being the maximum calculated risk). CMRS calculates a risk score for two risk factors (IAVM and STIG) with a score from 0 to 4,000.

III. REQUIREMENTS

A. SECURITY-FOCUSED CONFIGURATION MANAGEMENT

According to NIST SP 800-128, "Security-focused Configuration Management (SecCM) is the management and control of secure configurations for an information system to enable security and facilitate the management of risk" [30]. SecCM improves upon the configuration management process with the integration of security policies into an organization's existing CM process. The process flow diagram in Figure 10 shows the four SecCM phases for developing a SecCM process.

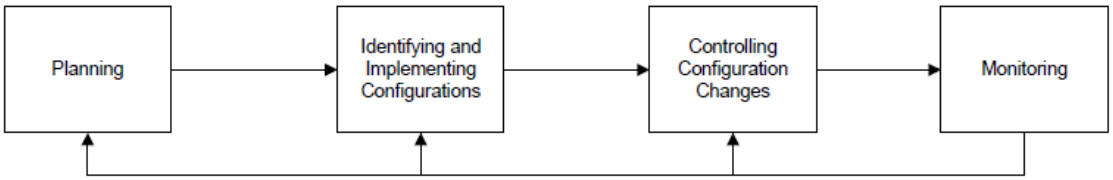


Figure 10. Security-Focused Configuration Management Phases

The configuration of a baseline for an asset is a component of the identifying and implementing Configurations phase of SecCM. An asset baseline can evolve over time but is established to provide a basis for future builds and changes to software and configurations. Creating and documenting the baseline configuration for an asset supports the implementation of NIST SP 800-53 control CM-2 baseline configuration [31].

1. Configuration Baseline Monitoring

An asset baseline configuration comprises the system specific security configuration that is required for the asset to function within its environment. The baseline configuration may include hardware components, software components, software configurations, operating system configurations, and documentation. An asset could have a different baseline configuration for each stage of its lifecycle.

As recommended by the NIST SP 800-128, "When possible, organizations employ automated tools to support the management of baseline configurations and to keep the configuration information as up to date and near real time as possible" [30]. Tools, such as group policy objects (GPOs) for MS Windows based servers, can be used to enforce a configuration baseline for an asset or group of assets. This automated method for providing policy enforcement can provide a degree of assurance that an asset is operating in a known secure state.

Issues can arise when relying solely on GPOs for maintaining a baseline if the management of these policies has not been incorporated into the CM process and undocumented changes are allowed that effect the enforced baseline. GPOs are limited in scope to the set of administrative templates that are available and may not cover all the required security settings in a configuration baseline. If a GPO fails to process due to an external issue, this can place the server in a non-compliant state that could go undetected if proper monitoring is not in place to detect these failures.

2. Secure Configuration Environment

As a best practice, organizations should validate security configuration baselines in an isolated environment before deploying to a production environment. As assets become more complex in function and rely on third party software and external components, the security configuration process becomes increasingly challenging. Many applications have specific operating requirements with functionality that can break down when a common secure baseline is applied. Isolation of assets, when building or modifying a configuration baseline, provides a controlled environment for testing configuration changes while protecting the production assets from the unsecured assets.

B. TRANSITION FROM VMS TO CMRS

The transition from VMS reporting to CMRS introduced unique challenges for managers of assets and an organization's standing accreditation. The scoring mechanism has changed substantially from the DoD severity codes used by Retina (reported to VMS) and the CVSS severity codes used by ACAS (reported to CMRS). There is not a one to one mapping between the severity codes from Retina to CVSS. The NVD provides severity rankings of high, medium, and low that mapped directly to the severity codes provided by Retina. To integrate support for CVSS scoring the NVD has mapped the CVSS numerical values to its existing severity codes, high (7.0 10.0), medium (4.0-6.9), and low (0.0-3.9).

Table 3 demonstrates the disparity between the severity codes reported by the legacy vulnerability assessment tool and the latest DoD tool.

Vulnerability Finding Variances between Retina & ACAS		
STIG Finding:	**Retina's DoD Severity Code**	**ACAS's CVSS Severity Code**
Microsoft HTML Help Buffer Overflow (Zero-Day)	CAT I	Info, CMRS Assigns a 0 severity for zero-day vulnerabilities
Password Does Not Expire	CAT II	Critical, CVSS = 10
Microsoft .NET Framework Multiple Vulnerabilities (2012-IAVA-001)	CAT I	Medium, CVSS = 6.8
Removable Disk (Detection of a USB Storage Device)	CAT IV	Critical, CVSS = 10
Allocate Floppy (Floppy Drive should be restricted to use only by currently logged in user)	CAT III	Critical, CVSS = 10

Table 3. Retina Versus ACAS Severity Code Comparison

Migrating a system to the ACAS / CMRS solution will undoubtedly result in a change to the reported and accredited risk assessment score. A DAA that has accepted the reported risk of an asset may require the reevaluation of an asset due to the change risk score in order to accept the newer assessment.

The current release of CMRS, as of August 16, 2013, is only capable of displaying a management/executive view of an organization's total risk assessment score based on the sum of all assets associated with that organization. A future release is planned to provide the ability to view individual asset assessments. The CMRS tool does not support the input of POA&Ms for findings associated with an asset and at this time there is no way of providing mitigation write-ups to lower a reported findings severity code recorded in CMRS [29]. As a result, the presence of false positives will skew the assessment data present in the system.

C. SYSTEM CONCEPT

A continuously monitoring/automated validation system that could fill some of the gaps identified above should be capable of several core functions. The system should be able to digest SCAP compliant validation reports, when available, and store scan results data within a database. It should have the ability to consume files on a regular basis through automated or manual actions, cataloging results by host, finding, definition, result, and time of scan, providing a near real-time view into each monitored asset's compliance state.

Many SCAP compliant tools already exist for server validation that provide results in a standard format that can be reliably parsed to obtain host and compliance data. Utilizing these pre-existing tools will avoid the need to develop an additional system component and allow an organization to continue to utilize their existing tools.

Integrating networking devices into this system will require creation of a validation component that is capable of parsing through flat configuration files completing STIG vulnerability checks and outputting compliance results. In order to support the wide range of networking devices and their applicable STIG checklists, the system must allow the creation of custom content that enables the scripting of checks for their applicable vulnerabilities. The system should support the ability to export the scripted checks and scan results. This capability would provide an organization the ability to run the scans from an external source. The resultant compliance data should be stored in a database capturing the device hostname, finding reference,

definition, compliance state and the time and date for the results data.

A major aspect of implementing SecCM involves the establishment of system baselines for each asset and applicable lifecycle state, as well as an isolated environment for testing configuration settings when building an asset's secured configuration. This means the tool must be capable of operating as a standalone system in environments dedicated to any stage of development. It must also allow users to track changes in the security baseline of a single host while supporting the ability to add notes specific to that system or assessment finding. This will provide users with the ability to justify open findings or enter notes specific to a system's baseline settings.

The sum of the these capabilities, along with the ability to operate without affecting a site's CMRS scores, show that the proof-of-concept system address some specific use cases that ACAS and CMRS do not.

Figure 11 is a conceptual diagram that illustrates the various functions and components of the proof-of-concept system. The sections that follow provide an overview of components required to assemble and develop this system.

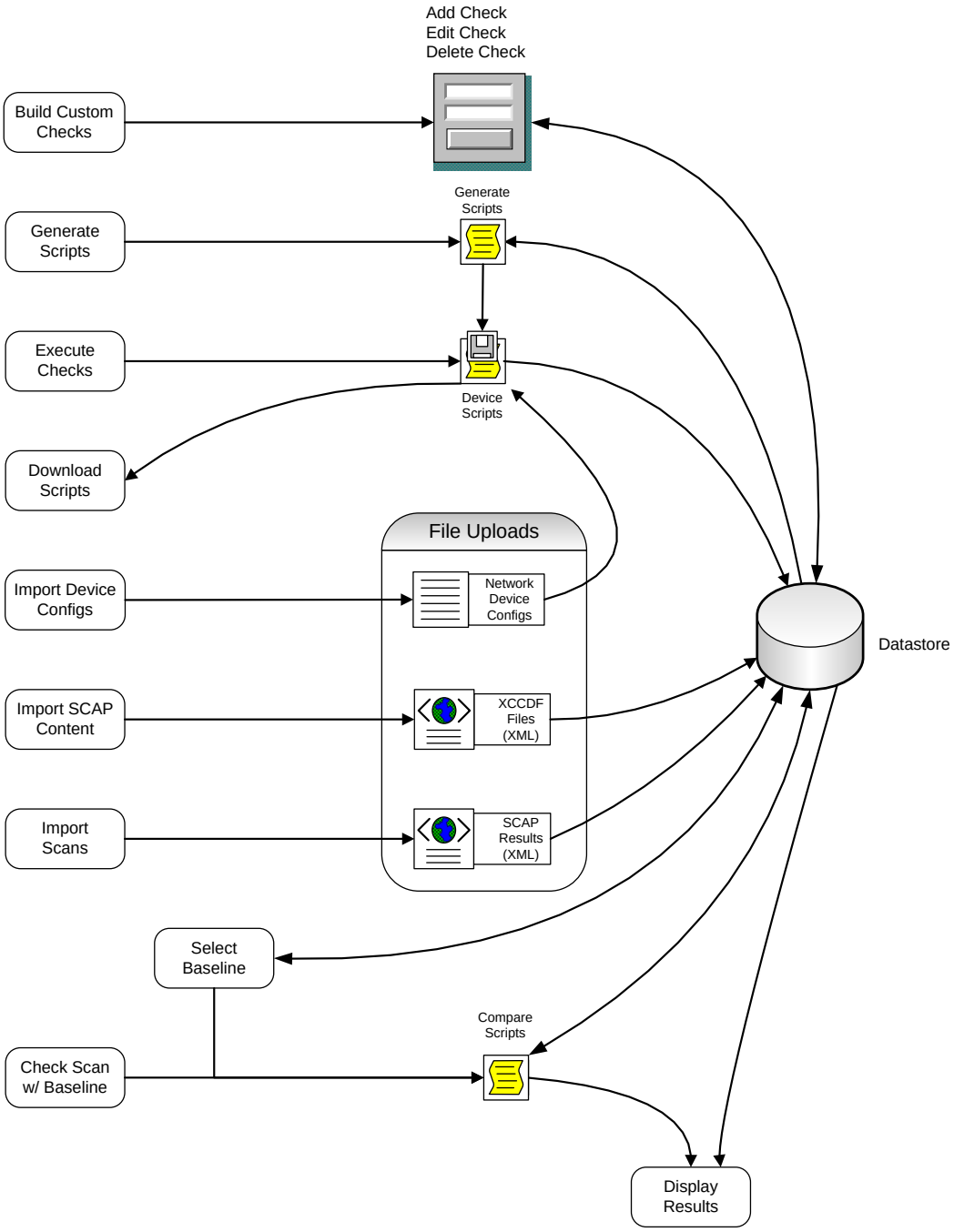


Figure 11. System Functional Diagram

1. Scripting Languages

The automated compliance validation system depicted in Figure 11 is predicated on having the ability to parse through various files. The three most critical types of

files are as follows: SCAP XCCDF files, which detail the definition ID, vulnerability ID, version number, category levels and titles of a structured set of security checks for some target system or component; SCAP XCCDF result files that detail the relevant target host identifier, the time of the evaluation, the SCAP definition ID and the SCAP check result (true/false); and finally, network device configuration files, which are basically flat files read into the running environment line-by-line during device boot-up detailing the device's settings. The ability to effectively parse through these files will allow the proof-of-concept system to extract user-defined data of interest.

On a movie set, a script provides simple instructions to each actor or actress detailing, in clear language, what they should say or how they should behave given a certain set of circumstances. Similarly, a computer script is a special type of program, a set of simple instructions, often in textual form that can automate a set of tasks given a certain set of circumstances. Usually these tasks are those that alternatively could be executed by a human operator one at a time. In the case of an automated compliance validation system, these one-by-one tasks should be automated through the use of one or more scripting languages.

The simplest types of scripting can be achieved via shell scripting. Bourne Again Shell (BASH) [32] is one of the most common Unix/Linux command line interfaces or "shells". It comes standard on most versions of Unix/Linux and MAC OS X, though ports of BASH exist for many other systems. While BASH can be utilized in the one-by-one

interactive mode described in the previous paragraph, it also has the ability to run a script of commands. This makes "programming" or scripting in BASH relatively easy. This is analogous to a batch file on a Windows-based system.

For the most part, each line of a script can be tested via the command line interface first. This allows those with less experience to build their scripts line-by-line instead of utilizing the iterative process of testing and troubleshooting each script as a whole. Another advantage to utilizing BASH scripting is that many commands and functions native to BASH are ideal for parsing, searching, comparing and manipulating text files. This ability is especially important when it comes to evaluating network device configurations against specific command line security checks. This type of scripting will also support user-defined checks, allowing a user to create custom configuration checks based on STIG guidance or configuration settings specific to their organization. While BASH scripting is a very versatile tool, the proof-of-concept system could also take advantage of alternative scripting languages that are particularly suited for certain tasks. One of these is Perl.

Perl is a dynamic programming, or scripting language developed in 1987 by Larry Wall to make report processing easier. As explained in *Beginning Perl*,

> many programmers assume that PERL is an acronym for Practical Extraction and Report Language. However perlfaq1—the documentation that shipped with Perl—sets the record straight:

> ... never write "PERL," because perl is not an acronym, apocryphal folklore and post-facto expansions notwithstanding. [33]

Since its inception, Perl has undergone many changes including the borrowing of powerful text processing facilities that allow for easy manipulation of text files from other languages, such as C and shell scripting. In its current revision, Perl is used in a myriad of applications that take advantage of its flexibility and coarse simplicity. What makes Perl so attractive to the proof-of-concept system is its use of regular expressions as explained by Sammy Esmail:

> It is no secret that Perl regular expressions are the envy of other languages. As data continues to have an ever-growing importance in today's world, regular expressions provide us with the power to slice and dice data so that we can measure, learn, and make intelligent decisions. Good regular expressions, such as those in Perl, will therefore become increasingly important. [34]

Perl's capabilities could augment the proof-of-concept system's ability to parse data by providing a way to parse data that may be in a format that might not be as suited for BASH scripts. Given Perl is open source, relatively easy to use because it favors language constructs that are natural for humans to understand, and runs on virtually any platform, Perl could be a very useful component of the proof-of-concept system.

While Perl is suitable, "PHP is the most popular server-side scripting language in web development, powering an estimated 78.9% of all websites" [35]. Originally developed in 1994 by Rasmus Lerdorf, these personal home page tools were a collection of small programs or scripts

used to maintain his website. Over the years, continued development by others has pushed the meaning of PHP to now stand for PHP hypertext processor [36].

PHP is ideal for the proof-of-concept system for several reasons. It works well with HTML, which would form the basis for interacting with the proof-of-concept system. It is also relatively easy to learn and has hundreds of built in functions and thousands more available through extensions, which makes is suitable for many tasks. Several of these built-in functions are particularly suited for dealing with XML files that could provide the basis for proof-of-concept the system's ability to process and consume much of the SCAP content available.

Finally, it is free and easy to install as part of the Apache/MySQL/PHP (AMP) [37] software stack on Linux, can run on virtually any web server, platform, or OS, and can interact with many relational database management systems (RDBMS). This last attribute allows the proof-of-concept system to take advantage of the inherent power of databases.

2. Relational Database

Another key requirement for this system is the ability to store data in an organized way so it can be searched and retrieved later. The relational database, pioneered by E. F. Codd in his 1970 paper, "A Relational Model of Data for Large Shared Data Banks," is ideally suited for storing, organizing and manipulating data. As summarized on *Wikipedia*:

> In relational databases, each data item has a row of attributes, so the database displays a fundamentally tabular organization. The table goes down a row of items (the records) and across many columns of attributes or fields. The same data (along with new and different attributes) can be organized into different tables. [38]

The characteristics of the relational database provide many potential applications for use in a compliance validation system. A system capable of consuming XCCDF and SCAP result files and entering this data into a relational database would have the ability to perform many tasks. With this information stored within a relational database, the system should be capable of processing, comparing and displaying information in many different ways. Among other things, this would allow for baseline comparisons reports and reports by individual vulnerability, finding, or server.

The most common means to take advantage of all a relational database has to offer is to utilize a relational database management system (RDBMS). An RDBMS is a software solution used to define, create, manage, query, and update relational databases. Nearly all RDBMS products available today are American National Standards Institute (ANSI)/International Organization for Standards (ISO) Structured Query Language (SQL) compliant [39]. As a result, any standards compliant SQL RDBMS can be used.

According to its website, MySQL is the world's most popular open source database, with over 65,000 downloads per day. This is partially due to it being a central component of the AMP software stack [37] that is often used in open source development projects. Larger projects, such

as Wikipedia, Facebook, Twitter, YouTube, and Flickr also rely on MySQL but are most likely utilizing a paid, more feature-rich version.

3. Front End Web Server

Another key requirement for the proposed system is a graphical user interface (GUI). This portion of the system allows users to upload, create, modify, delete, and view content/data. While a traditional, software-defined GUI would meet these needs, utilization of a web front-end allows almost any user with an EUD to interact with the system.

The two most popular options, those with the highest market share among all websites as noted in Netcraft's December 2013 web server survey, are the Apache (41 percent) and Microsoft (28 percent) offerings while the balance is split between nginx (15 percent) and Google (four percent) [40]. Besides being the most popular web server software in the world, Apache offers several advantages over the other choices.

Apache is open source and can run on virtually any of the commonly used operating systems. This provides some flexibility that MS Internet Information Services (IIS) does not. For example, the current version of IIS is only supported on MS Windows Vista, MS Windows 7, and MS Windows server variants, which normally require a licensing fee to be paid [41]. Apache's ability to run on nearly any OS allows users of the system to install it practically anywhere. With Apache, the proof-of-concept system could be run on a MS Windows based laptop, a MS Windows based server

or virtually any Linux or Unix OS providing flexibility that is just not possible with IIS.

Apache is also part of the AMP software stack. As part of this software stack, is it easily installed as part of a precompiled package available from most mainstream Linux distributions where it is referred to as LAMP (Linux-AMP). For non-Linux OS, install packages can be downloaded from the Apache HTTP Server project website [42]. Additional features include secure sockets layer (SSL), transport layer security (TLS), authentication modules, and common language interface support for Perl, Python, and PHP.

As the largest software company in the world, Microsoft provides potential hackers with the greatest number of potential victims and therefore Microsoft products provide the biggest "bang for the buck" for cyber-criminals. By steering clear of IIS, a whole host of potential exploits can be avoided. Of course, any product will have its share of vulnerabilities and respective updates; it is the responsibility of the system owner to maintain proper levels of security.

4. Additional Concerns

There are several additional security items of concern that relate to the functions and components described in the previous sections. One of the first is the ability to control who has access to the system. While the proof-of-concept system concentrates on its core functions, it is important to mention that role based access controls (RBAC) [31] could be used to limit access to various components of the system to those with an appropriate administrative role.

Another concern is the ability to control what can be uploaded or imported into the system. This particular item addresses two different scenarios. The first is the ability to perform some type of input validation during file uploads. This should help prevent someone from maliciously uploading an inappropriate file or prevent a user with good intentions from simply uploading an incorrect file type.

The second scenario addresses what type of information should be imported into the system. For example, if the system's primary function is to store validation results necessary baseline comparisons, there would be no need to store entire device configurations within the database. Doing so would needlessly introduce potentially sensitive data into the system offering an additional exploitation vector.

Finally, a whole host of STIG and security settings must be applied to the proof-of-concept system itself. A system used to validate and track asset compliance should be held to even higher standards of security than many, if not all, of the systems it is tracking so that the system components do not negatively affect the overall risk of an organizations assets. Based on some of the components describe above, several checklists, including those for OS, Database, and Webserver, are at a minimum applicable to the proof-of-concept system detailed in the next chapter.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. PROOF-OF-CONCEPT SYSTEM

As detailed in the previous sections, many different components could have been used to develop the proof-of-concept system. For the purposes of this development effort, a web front end, a database, and at least one scripting language are required. When evaluating the various options, it is clear that a Linux based host using the LAMP software stack provides the most convenient development system. As an added bonus, Linux's built in support of BASH allows shell scripting to be utilized without additional modifications.

For this particular effort, an Apache name-based virtual host website was configured on a shared Cent OS Linux server. An Apache name-based virtual host allows for the hosting of multiple web sites on a single internet protocol (IP) address. This particular server was hosted on a consumer grade internet connection and was remotely accessible via secure shell (SSH) using a private key/public key exchange for authentication. By hosting the development site on the internet, each member of the team could work collaboratively or on their own while maintaining all code in a central location.

Many factors play a role in an organization's selection of system components. The use of Apache and MySQL are appropriate in this case, but an organization that relies on other compatible products could easily decide to utilize Microsoft's IIS and Microsoft's SQL software if these components are preferred.

A. INDIVIDUAL FUNCTIONS

The rest of Chapter IV primarily details how the system functions from a user's perspective. The bulk of this interaction is through the web interface, which consists of a basic menu of tabs for each of the system's core functions. The individual tabs or functions are described in the following sub-sections. The database tables and data types used in this proof-of-concept system are found in Appendix A. The various supporting code source files used are found in Appendix B.

1. Import

The import tab/function checks for XCCDF XML files in a folder named "content" in the website root folder. The Import tab webpage is generated based on the files in the content folder. If the XCCDF XML file has not been imported into the database, an import button is available for that file and is selected to import the XCCDF content. A sample view of the import content table is shown in the Figure 12.

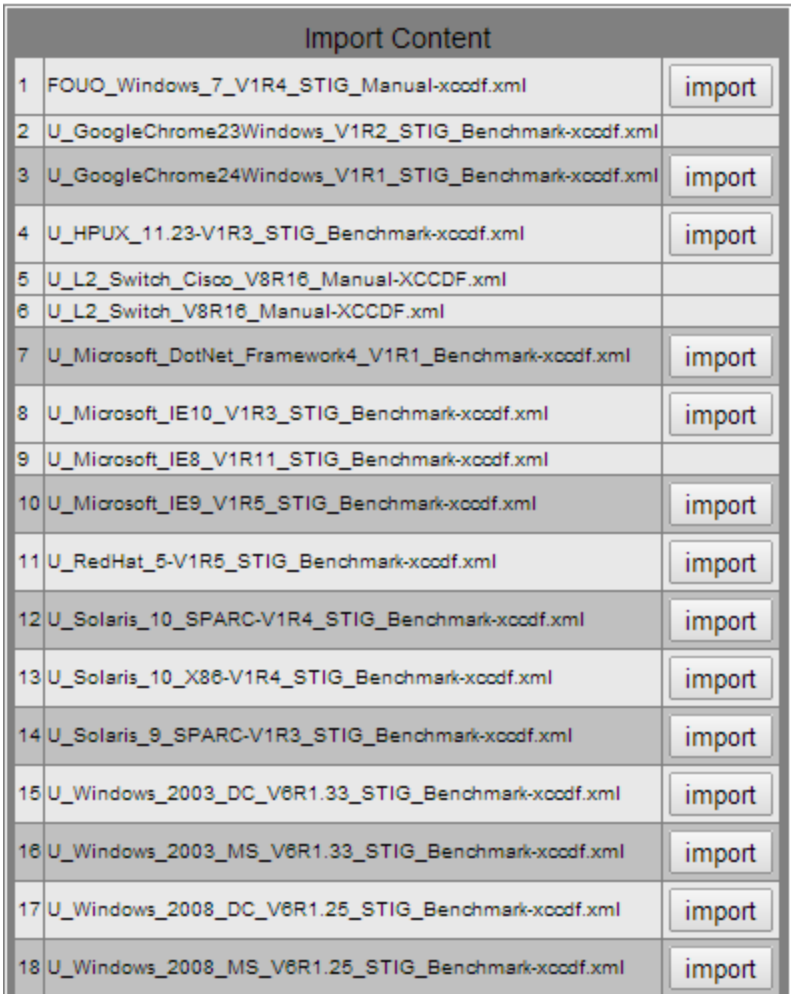


Figure 12. Import XCCDF Content

The import function parses through an XCCDF Manual or Benchmark file. A manual XCCDF file contains all the checks associated with a platform or application STIG. A benchmark XCCDF file contains only automated SCAP checks and SCAP definition data. The import function parses data from these files and stores this data in the database to be utilized by other system functions.

2. Codefunctions

The code functions tab/function allows the user to create a snippet of code to be used as a template when

creating specific checks in the Groups tab. Shell based code can be entered into the code section and saved along with various other attributes, such as name, description, and creator. Figure 13 shows where code can be created and added.

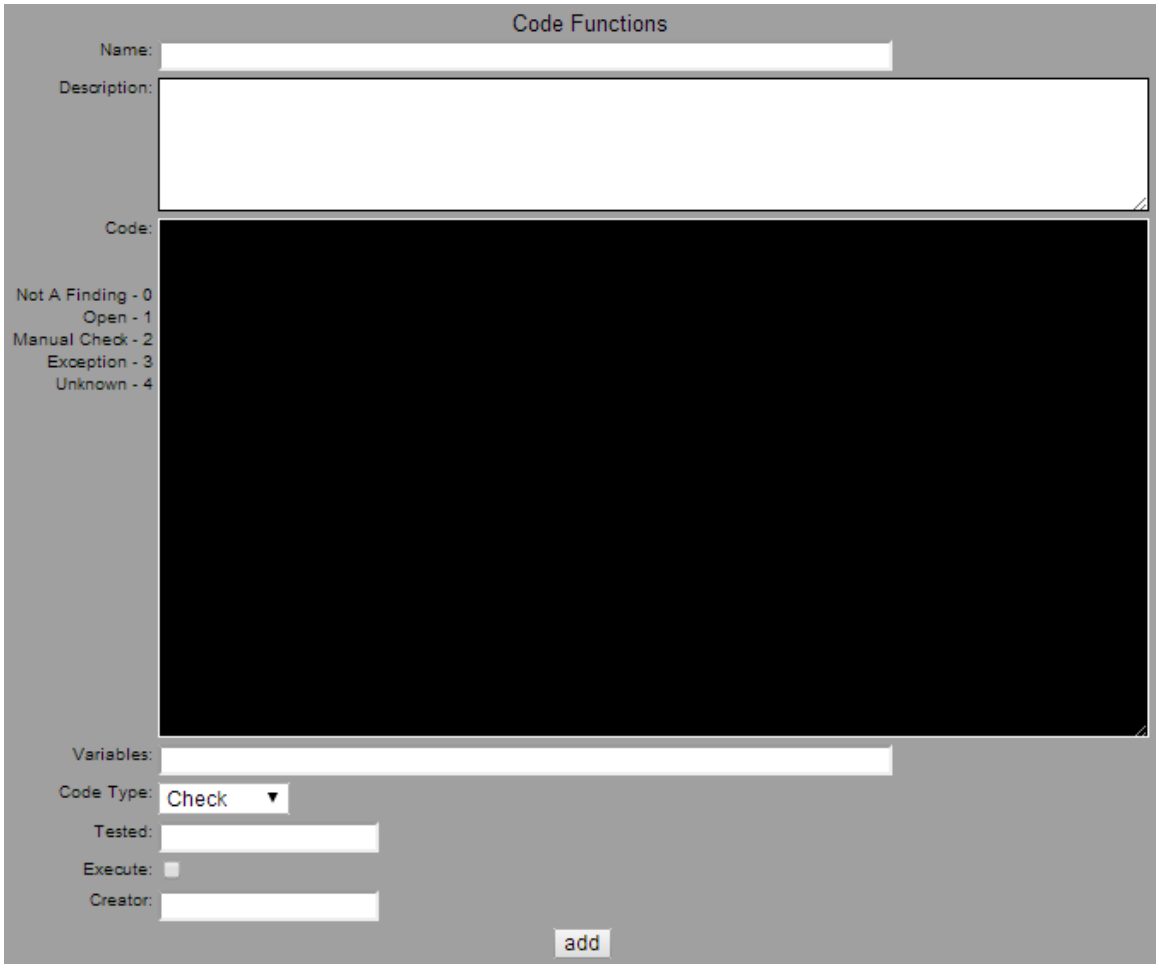


Figure 13. Create Code Functions

In order to edit or delete a code function, the user first has to select the template by clicking on the name in the table shown at the bottom of the Code Functions page. This table is shown in the Figure 14. After a template is selected, the main area of the page is populated.

Name	Type
1 Null Output is Good	Check
2 Null Output is Bad	Check
3 BASH Template	Template
4 Manual Check	Check
5 Cisco Config Null is Bad	Check

Figure 14. Code Functions List

From here the user has the option to delete or make changes to the existing code function. In Figure 15, the code function "Cisco Config Null is Bad" has been selected. If check returns no output, the check is considered to have failed.

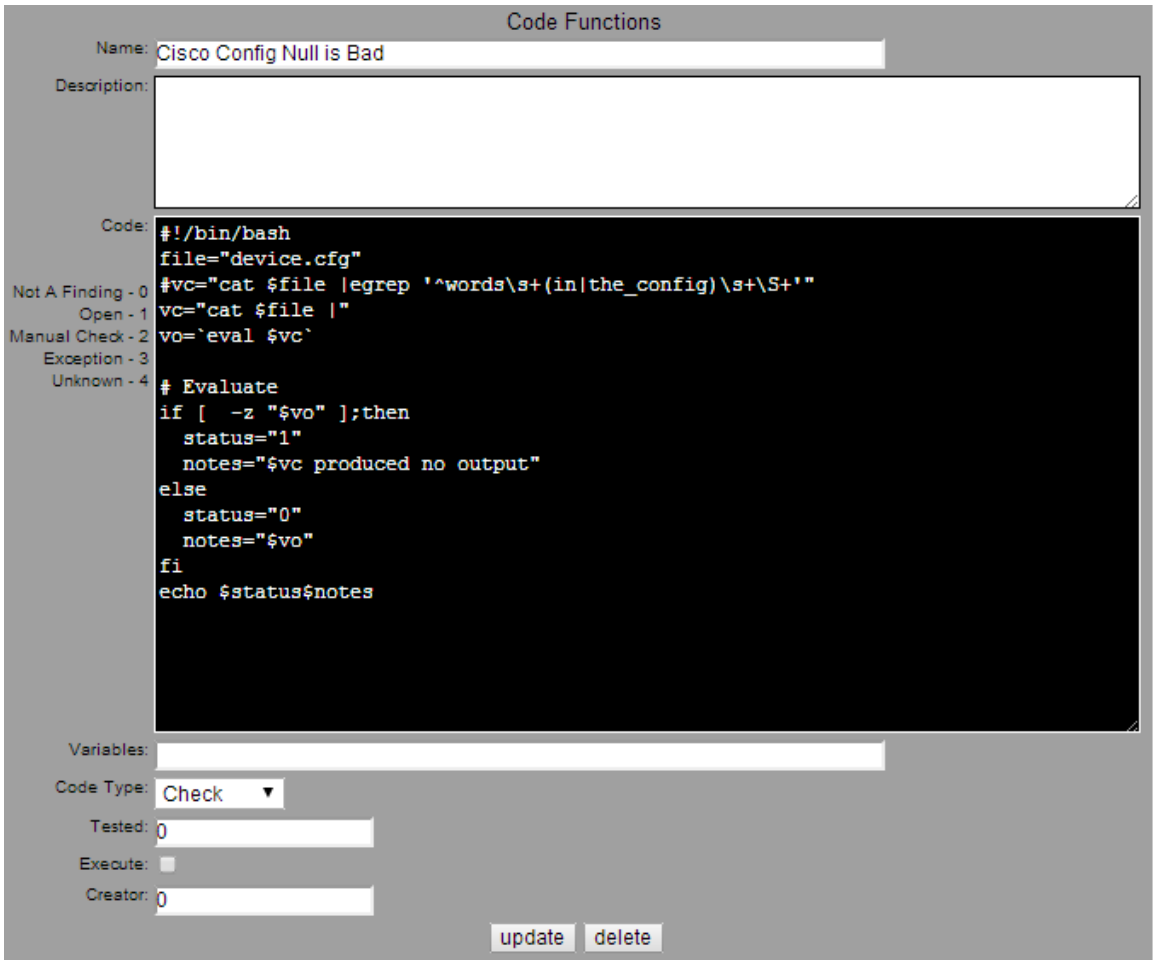


Figure 15. Edit Code Functions

In this case, the function is checking for specific configuration commands within the device configuration file, "device.cfg". If it finds specific configuration commands, the status is set to "0," which is passing. The proof-of-concept system then displays, within the notes, the specific line found in the configuration. If the configuration commands are not found, the status is set to "1," which is failing, and the code specific to the check is concatenated with the words "produced no output" to clearly indicate exactly the commands that were executed and that nothing was found. From this same interface

changes are made and saved or the entire code function is deleted, using the update and delete buttons respectively.

3. Documents

The documents tab/function displays the current list of XCCDF XML files that have been imported in the database. The XCCDF files and their document titles are displayed in a table similar to the one shown in Figure 16.

DocumentTitle	Xmlfile	
Windows Server 2008 R2 Member Server Security Technical Implementation Guide	U_Windows_2008_R2_MS_V1R9_STIG_Manual-xccdf.xml	select
Layer 2 Switch Security Technical Implementation Guide - Cisco	U_L2_Switch_Cisco_V8R16_Manual-XCCDF.xml	select
Layer 2 Switch Security Technical Implementation Guide	U_L2_Switch_V8R16_Manual-XCCDF.xml	select
Google Chrome v23 Windows STIG	U_GoogleChrome23Windows_V1R2_STIG_Benchmark-xccdf.xml	select
Internet Explorer 8 STIG	U_Microsoft_IE8_V1R11_STIG_Benchmark-xccdf.xml	select
Internet Explorer 9 Security Technical Implementation Guide	U_Microsoft_IE9_V1R5_STIG_Benchmark-xccdf.xml	select

Figure 16. XCCDF Documents List

Clicking the select button brings up a table that displays all applicable profiles associated with the selected XCCDF document. Each profile contains a list of applicable findings associated with that profile's classification and mission assurance category (MAC) level as shown in the Figure 17.

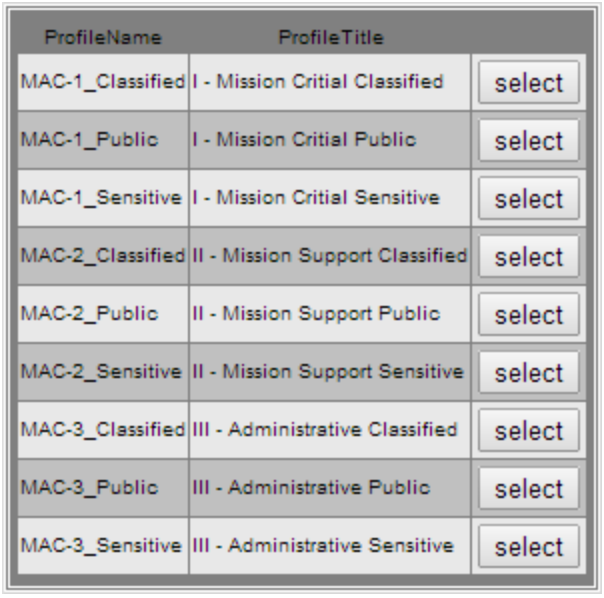
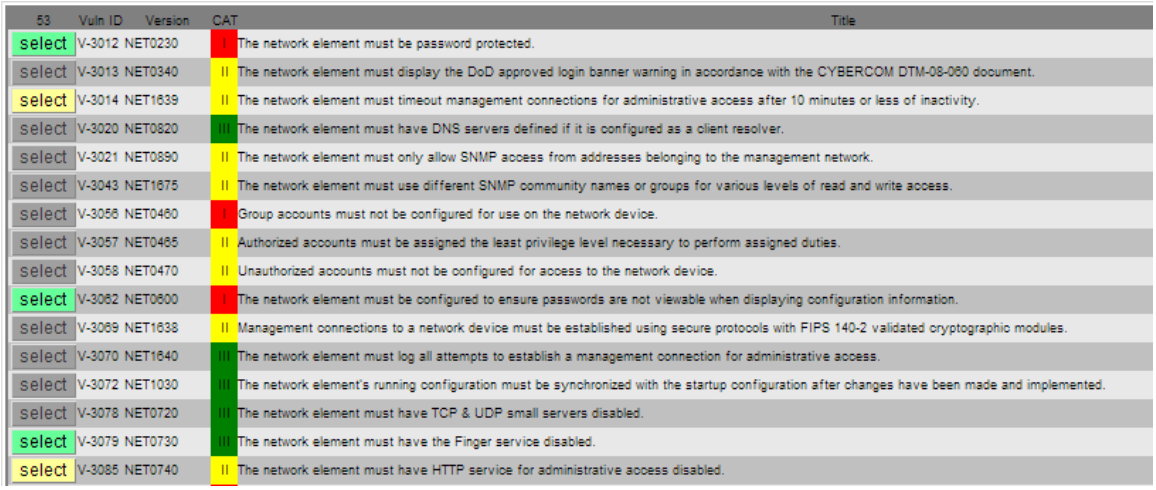


Figure 17. Select Profile

Selecting a profile loads the findings from the database to a table that is viewed from the groups tab.

4. Groups

The groups tab/function displays the individual vulnerabilities associated with a particular profile. As shown in Figure 18, the count, vulnerability ID, version, CAT level, and title are all displayed.



53	Vuln ID	Version	CAT	Title
select	V-3012	NET0230	I	The network element must be password protected.
select	V-3013	NET0340	II	The network element must display the DoD approved login banner warning in accordance with the CYBERCOM DTM-08-060 document.
select	V-3014	NET1639	II	The network element must timeout management connections for administrative access after 10 minutes or less of inactivity.
select	V-3020	NET0820	III	The network element must have DNS servers defined if it is configured as a client resolver.
select	V-3021	NET0890	II	The network element must only allow SNMP access from addresses belonging to the management network.
select	V-3043	NET1675	II	The network element must use different SNMP community names or groups for various levels of read and write access.
select	V-3056	NET0460	I	Group accounts must not be configured for use on the network device.
select	V-3057	NET0465	II	Authorized accounts must be assigned the least privilege level necessary to perform assigned duties.
select	V-3058	NET0470	II	Unauthorized accounts must not be configured for access to the network device.
select	V-3062	NET0600	I	The network element must be configured to ensure passwords are not viewable when displaying configuration information.
select	V-3069	NET1638	II	Management connections to a network device must be established using secure protocols with FIPS 140-2 validated cryptographic modules.
select	V-3070	NET1640	III	The network element must log all attempts to establish a management connection for administrative access.
select	V-3072	NET1030	III	The network element's running configuration must be synchronized with the startup configuration after changes have been made and implemented.
select	V-3078	NET0720	III	The network element must have TCP & UDP small servers disabled.
select	V-3079	NET0730	III	The network element must have the Finger service disabled.
select	V-3085	NET0740	II	The network element must have HTTP service for administrative access disabled.

Figure 18. XCCDF Vulnerability List

Each vulnerability has a select button associated with it. These buttons appear in several colors. The default color is grey, and upon initial import, all vulnerabilities begin with this color. Green buttons indicate that the check's status has been marked as tested. If a check has been marked as having a bug, meaning the check does not function properly, the button is red. Finally, yellow buttons indicate that the check has been added, but it has not been marked as tested or as having a bug.

When selecting one of the vulnerabilities, the user is presented with an interface to create a custom check. The user may choose to import one of the previously defined code functions by selecting one from the drop-down and inserting the template code into the coding area. Alternatively, the user may type directly into the coding area. In either scenario, the user has the ability to customize the script as needed. Figure 19 displays the custom check for verifying that a password has been set on a Cisco Switch or Router. This particular check was created using the "Cisco Config Null is Bad" template.

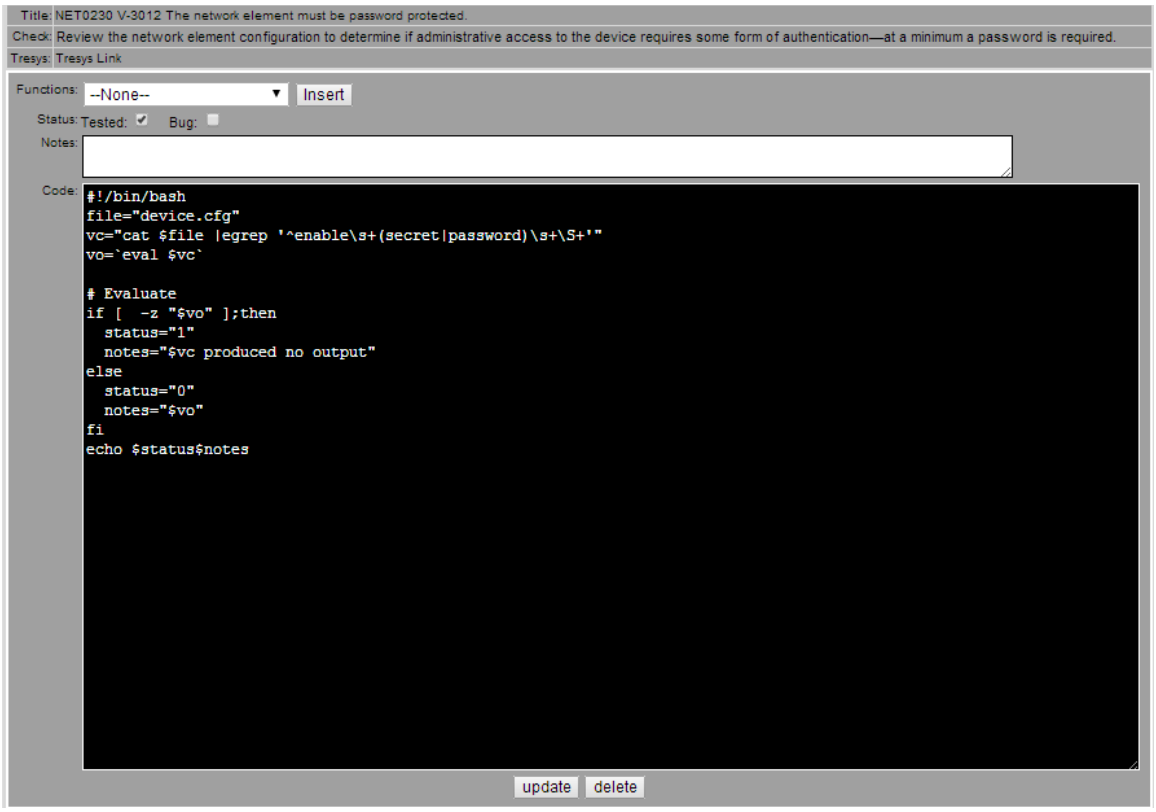


Figure 19. Create Custom Check

Each custom check created, is stored within the database associated with that particular vulnerability. These checks are used by the generate scripts function.

5. Generate Scripts

A configuration file needs to be selected from the config tab, before the generate script tab is visible. The generate scripts tab/function creates scripts from all the custom checks created in the documents tab. Once the generate scripts tab is selected, the scripts are generated, compressed, and stored in a tape archive (TAR) file. Figure 20 shows sample display output from the generate scripts function from five custom network checks.

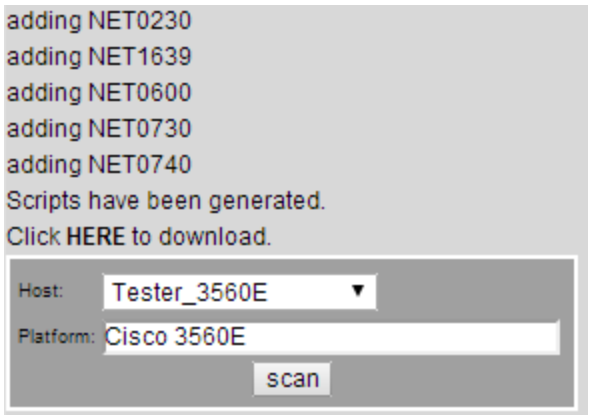


Figure 20. Generate Custom Scripts

Selecting HERE from "Click HERE to download" allows the user to download the TAR file containing all the custom shell scripts for use on a standalone EUD. If utilizing the proof-of-concepts scan function, the user can select a host, and name the associated platform for the configuration. Selecting the scan button runs the scripts against the selected configuration file and produces an output similar to the one shown in Figure 21.

```
RULE ID: SV-3012r2_rule   VULN ID: V-3012   VERSION: NET0230   STATUS: 0
TITLE: The network element must be password protected.
NOTES: enable secret 5 $1$vGR1$97E/Oqw4XXXXXXXXr3ms1

RULE ID: SV-3013r2_rule   VULN ID: V-3013   VERSION: NET0340   STATUS:
TITLE: The network element must display the DoD approved login banner warning in accordance with the CYBERCOM DTM-08-060 document.
NOTES:

RULE ID: SV-41449r2_rule   VULN ID: V-3062   VERSION: NET0600   STATUS: 0
TITLE: The network element must be configured to ensure passwords are not viewable when displaying configuration information.
NOTES: username user1 privilege 0 secret 5 $1$YWQC$PTXXXXXXXXXlBMUbIrfMu0

RULE ID: SV-15305r2_rule   VULN ID: V-3079   VERSION: NET0730   STATUS: 1
TITLE: The network element must have the Finger service disabled.
NOTES: cat device.cfg |egrep '^no ip finger|no service finger' produced no output

RULE ID: SV-41467r1_rule   VULN ID: V-3085   VERSION: NET0740   STATUS: 0
TITLE: The network element must have HTTP service for administrative access disabled.
NOTES: no ip http server
no ip http secure-server
```

Figure 21. Execute Custom Scan

The output from the scan displays the rule ID, vulnerability ID, version, status, title, and notes associated with the custom check. The STATUS out provides

the findings current compliance state if there is one (0 = passing, 1 = failing).

6. Hosts

The hosts tab/function allows the user to add hosts that are linked to scan results. The hosts created are identified by data entered in the name and description fields. Once a host is added, it appears in the table like the one shown in Figure 22.

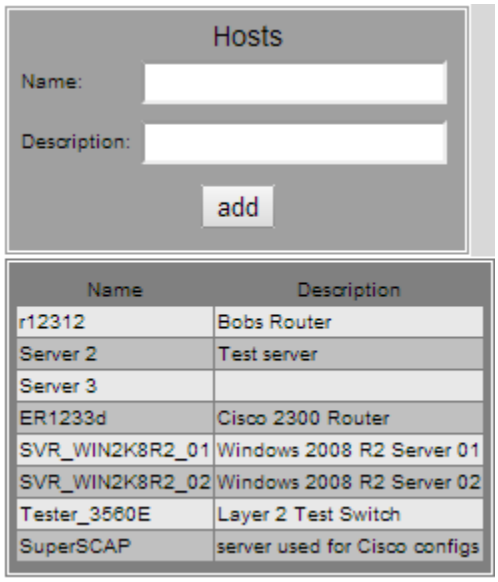


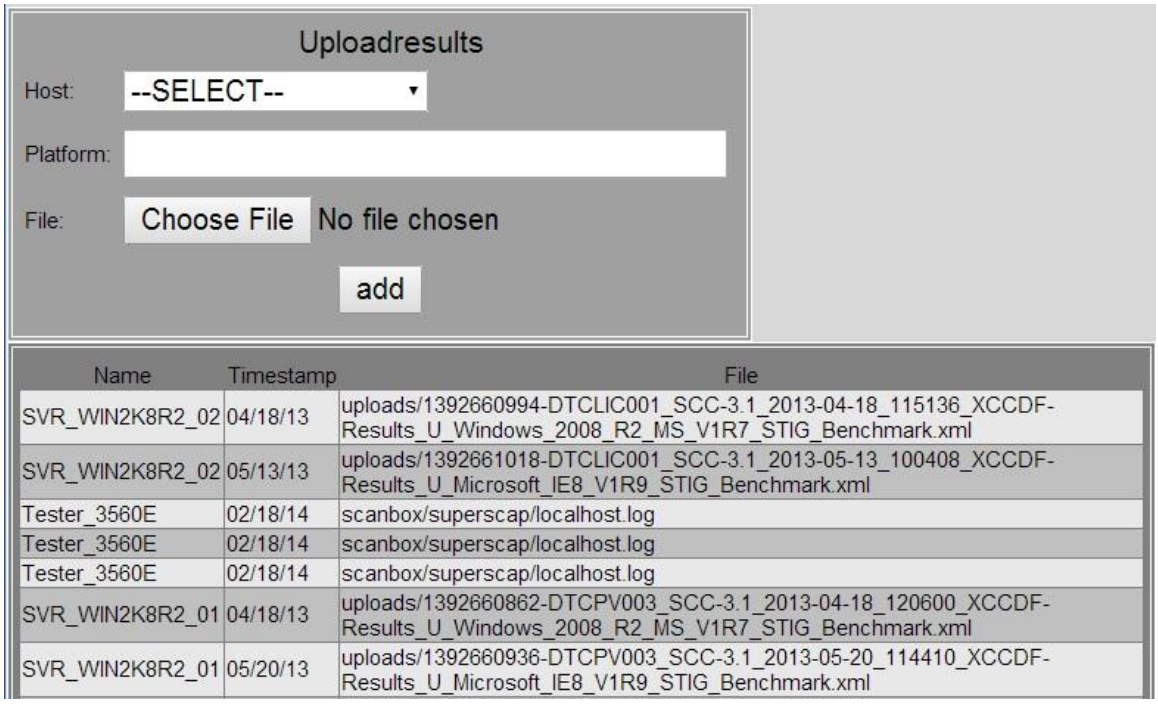
Figure 22. Add Hosts

The host data input is stored in the Hosts table of the database. Selecting a host allows the user to edit the host's data or delete the host.

7. Uploadresults

Through the upload results tab/function the user uploads XCCDF results files generated from SCAP compliant tools. The uploaded files are stored in the directory

"uploads," created under the website root folder. The platform field is used to specify the platform or application associated with the uploaded scan result. A listing of uploaded XCCDF results files and network scan results is shown in Figure 23.



Name	Timestamp	File
SVR_WIN2K8R2_02	04/18/13	uploads/1392660994-DTCLIC001_SCC-3.1_2013-04-18_115136_XCCDF-Results_U_Windows_2008_R2_MS_V1R7_STIG_Benchmark.xml
SVR_WIN2K8R2_02	05/13/13	uploads/1392661018-DTCLIC001_SCC-3.1_2013-05-13_100408_XCCDF-Results_U_Microsoft_IE8_V1R9_STIG_Benchmark.xml
Tester_3560E	02/18/14	scanbox/superscap/localhost.log
Tester_3560E	02/18/14	scanbox/superscap/localhost.log
Tester_3560E	02/18/14	scanbox/superscap/localhost.log
SVR_WIN2K8R2_01	04/18/13	uploads/1392660862-DTCPV003_SCC-3.1_2013-04-18_120600_XCCDF-Results_U_Windows_2008_R2_MS_V1R7_STIG_Benchmark.xml
SVR_WIN2K8R2_01	05/20/13	uploads/1392660936-DTCPV003_SCC-3.1_2013-05-20_114410_XCCDF-Results_U_Microsoft_IE8_V1R9_STIG_Benchmark.xml

Figure 23. Upload Results

The table generated displays the list of all the XCCDF results uploaded, including the host name, timestamp the scan was completed, and the XCCDF results file name. The table also displays network scan results that have been automatically imported into the results database as a result of initiating a scan from the generate scripts tab.

8. Uploadconfig

The upload config tab/function allows the user to upload a device configuration file to the proof-of-concept

system. The uploaded files are stored in the directory "uploads" created under the website root folder. The user selects the host associated with the device configuration file and provides a description for the uploaded configuration file. The upload config page looks similar to Figure 24.

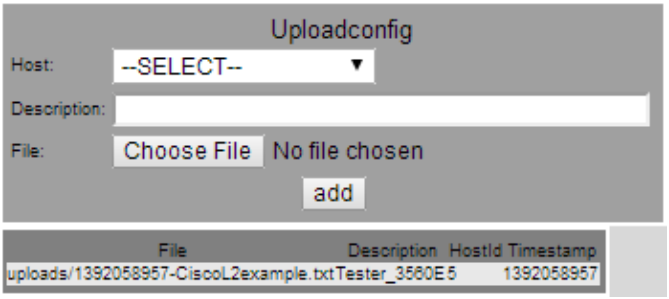


Figure 24. Upload Config

This table provides a list of all the configuration files uploaded, including the user provided description, host ID, and timestamp.

9. Scans

The scans tab/function displays a list of all the XCCDF results uploaded and the network device scan results generated from the custom check scripts. Figure 25 displays the table showing the date the scan was completed, the host's name, the associated platform, and the scan results filename.

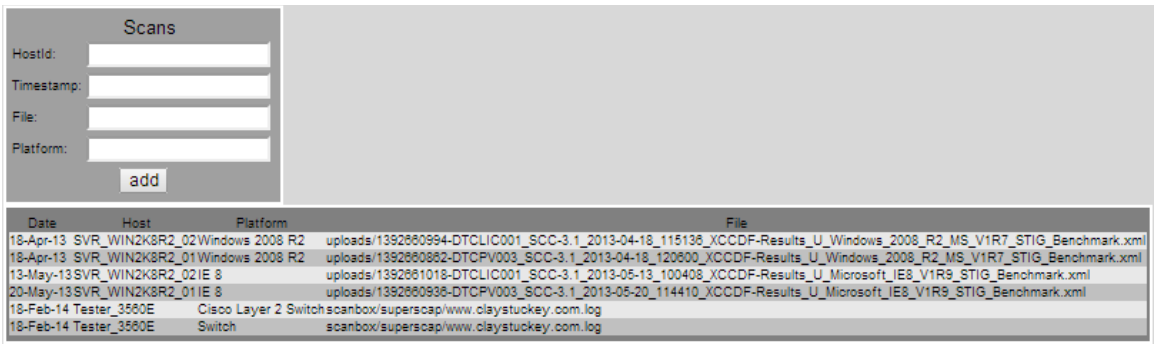


Figure 25. Scan Results

Selecting a scan result allows the properties associated with the scan to be modified or deleted.

10. Configs

The configs function/tab displays all the previously uploaded network device configuration files as shown in Figure 26. When a user selects a configuration file, the configuration file is written into the scanning directory and renamed device.cfg. The previously created custom checks are used to validate security settings.

Figure 26. Configuration List

When a user clicks on the file name for an individual configuration that file's location and name, description, host Id and timestamp are loaded into their respective fields. The device configuration is also loaded in the text field. The user can review the configuration manually as well as make changes the configuration's editable fields. The user can also delete the configuration. A truncated example of a loaded configuration file can be seen Figure 27.

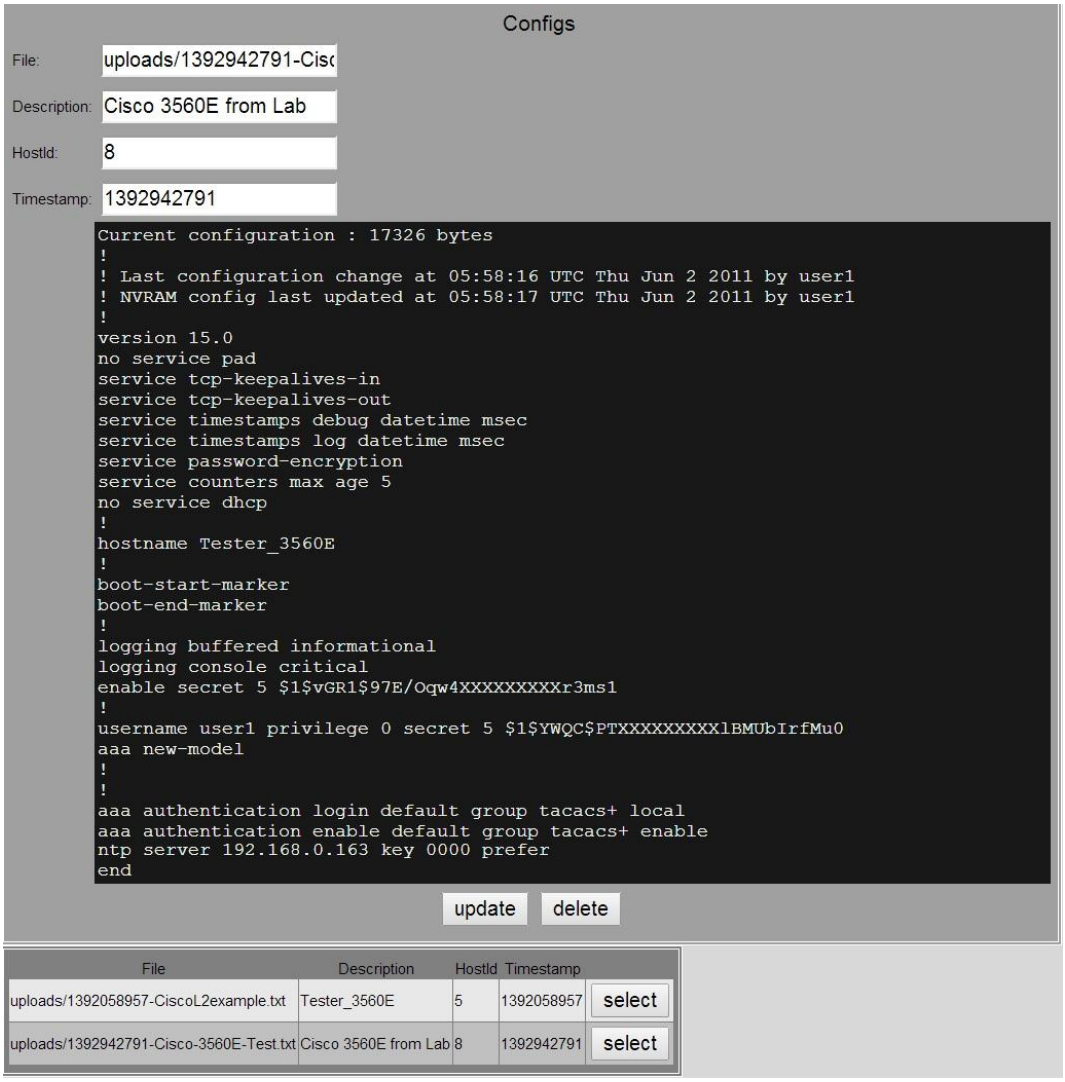


Figure 27. Loaded Configuration

11. Reviewscans

The review scans tab/function provides a list of all the scan results. The user selects a scan as the base line and a scan as the target as shown Figure 28.

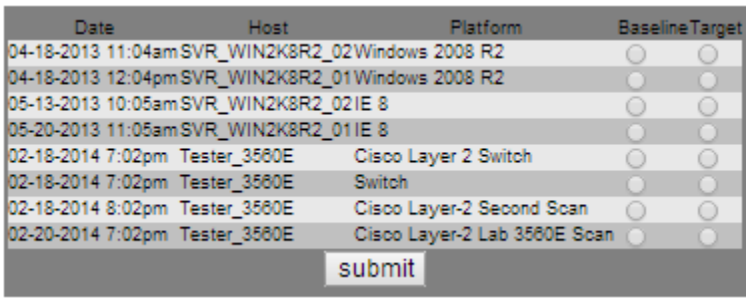


Figure 28. Review Scans Listing

The user clicks the submit button to compare the target scan result with the baseline. The provided output is a status of all the findings for the baseline and target scans. Non-matching results appear highlighted in red as shown in the Figure 29.

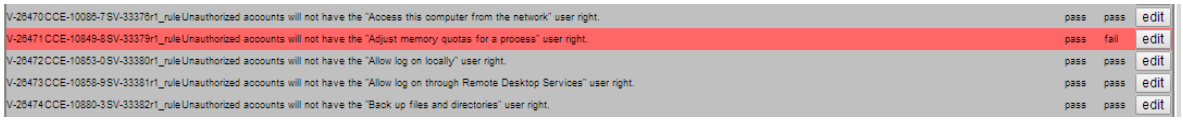


Figure 29. Review Scan Results

Selecting the edit button next to a finding brings up the results form, as shown in Figure 30. This allows the user to make changes or add notes to a specific result. This field is used for custom notes regarding false positives, POA&Ms, or simply for informational purposes.

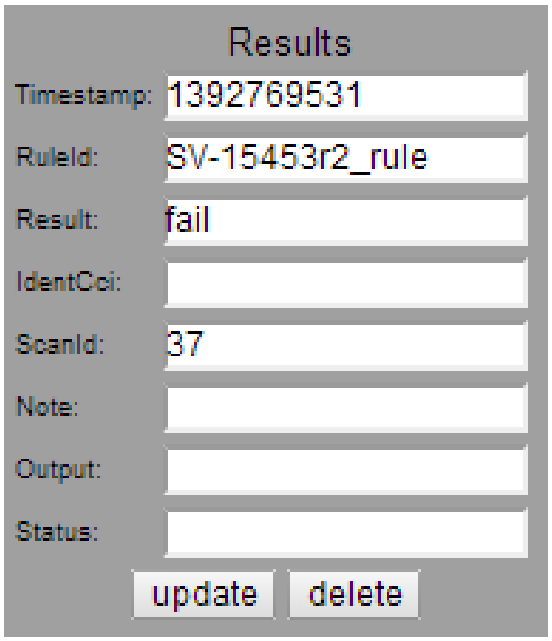


Figure 30. Modify Scan Result

Selecting the update button saves the user provided input, while the delete button will remove the finding from the database.

B. SYSTEM FLOW

To summarize the system flow, the following example is given. If a user wants to evaluate a network configuration and the XCCDF file for the evaluated asset is already imported; the user takes the following actions.

First, the user selects the host tab to add the device as a host, if it does not already exist. Then, the user selects the upload configs tab and uploads the config to be evaluated. The config is uploaded and the user selects the config tab to view the configs and selects the config to be evaluated. Next, the user selects the documents tab to view the list of XCCDF content imported into the system database and selects the applicable content for the network device,

as well as its profile (MAC level / sensitivity) for the operating environment. Once the document and profile have been selected, the groups and generate scripts tab appear, and the user is redirected to the groups tab. The user then edits any custom checks or views applicable STIG content if desired. Next, the user selects the generate scripts tab that creates the server-side custom check scripts. Finally, the user selects the host, provides the device platform and clicks the scan button to run the generated scripts against the selected device configuration file. The output from the checks is then displayed below the scan button for the user to read, and the results are entered into the database for use by other functions.

THIS PAGE INTENTIONALLY LEFT BLANK

V. FUNCTIONAL TESTING

In order to understand the proof-of-concept system's viability as an IA tool, it had to be put through functional testing. This testing was completed using actual SCAP benchmark data and actual network configuration files. The following sections detail the process and results of that testing in a step-by-step manner.

A. SERVER FUNCTIONAL TESTING

In order to validate server functionality, test data sets had to be created. A Windows 2008 R2 test server, SVR01_WIN2008R2, was used to generate SCAP benchmark data for testing the functional code for the proof-of-concept system. The SCAP Compliance Checker (SCC) tool, version 3.1, was used for generating the SCAP benchmark results files. The SCC tool was created by and maintained by Space and Naval Warfare (SPAWAR) Systems Center ATLANTIC [43]. The SCAP content for Windows Server 2008 R2 and Internet Explorer 8 were used to evaluate the STIG compliance of the test server. A preliminary scan was completed to produce a benchmark scan result file for the test server. Configuration changes to the base OS and Internet Explorer 8 were made to create a modified system target. These changes were made to bring these configuration settings out of compliance on the test server and are presented in Table 4.

Platform	Vuln ID	Rule ID	Description
Windows 2008 R2	V-1093	SV-32283r1_rule	Anonymous enumeration of shares will be restricted.
Windows 2008 R2	V-1102	SV-32287r1_rule	Unauthorized accounts will not be granted the "Act as part of the operating system" user right.
Internet Explorer 8	V-3428	SV-25181r1_rule	Internet Explorer is configured to allow users to change policies.
Internet Explorer 8	V-3429	SV-25180r1_rule	Internet Explorer is configured to allow users to add/delete sites.
Internet Explorer 8	V-6249	SV-25618r1_rule	The Java Permissions is not set properly for the Internet Zone

Table 4. Test Server Configuration Changes Modified

The modified system target was re-evaluated with the SCC tool generating a second set of benchmark scan result files. In Table 5, the filenames of the scan result files for the test server and the result type are shown.

Filename	Platform	Result Type
Baseline-SVR01_WIN2008R2_SCC-3.1_2013-06-03_115136_XCCDF-Results_U_Windows_2008_R2_MS_V1R7_STIG_Benchmark.xml	Windows 2008 R2	Baseline System
Baseline-SVR01_WIN2008R2_SCC-3.1_2013-06-03_115136_XCCDF-Results_U_Microsoft_IE8_V1R9_STIG_Benchmark.xml	Internet Explorer 8	Baseline System
SVR01_WIN2008R2_SCC-3.1_2013-06-29_101215_XCCDF-Results_U_Windows_2008_R2_MS_V1R7_STIG_Benchmark.xml	Windows 2008 R2	Modified System
SVR01_WIN2008R2_SCC-3.1_2013-06-29_115136_XCCDF-Results_U_Microsoft_IE8_V1R9_STIG_Benchmark.xml	Internet Explorer 8	Modified System

Table 5. Test Server SCAP Benchmark Result Files

1. Import XCCDF Content Files

The user uploaded the XCCDF content files, listed in Table 6, to the content directory of the web server.

Filename	Platform
U_L2_Switch_Cisco_V8R16_Manual-XCCDF.xml	Cisco L2 Switch
U_Microsoft_IE8_V1R11_STIG_Benchmark-xccdf.xml	Internet Explorer 8
U_Windows_2008_R2_MS_V1R11_STIG_Benchmark-xccdf.xml	Windows 2008 R2

Table 6. XCCDF Content Files

Once the files have been uploaded the user clicked the import tab. The list of xml files from the content directory is shown in the import content folder as seen in Figure 31.

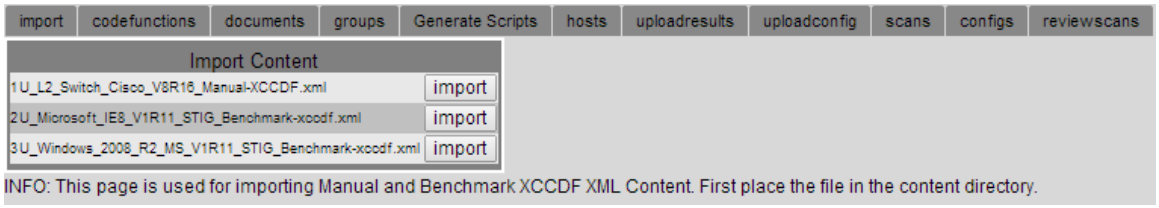


Figure 31. Import XCCDF Content

The user clicked the import button for each of the content files. After each of the files was imported, the import content table changed as shown in Figure 32.

Figure 32. Imported XCCDF Content

Once the import was completed, the user had the option to select the documents tab to see a list of the imported XCCDF content. This content would be used later, and reviewed during the testing of the network device functionality.

2. Adding Server and Network Device Hosts

Prior to uploading the scan result files for evaluation in the proof-of-concept tool, the user had to create an entry for the test server. To add a new host, the user selected the host tab, which brought up the "Add Hosts Dialog" shown in Figure 33.

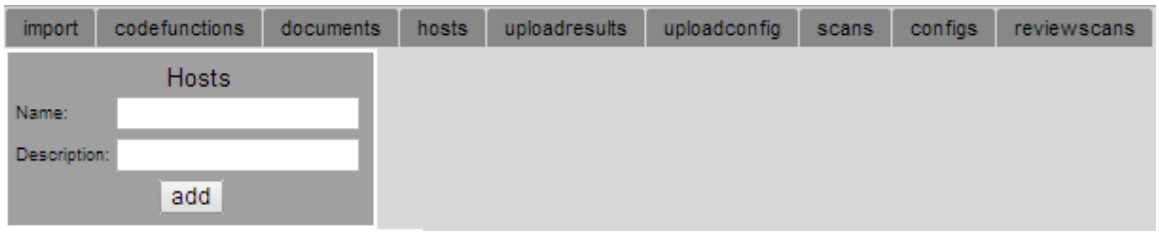


Figure 33. Add Hosts Dialog

The user entered the server hostname, SVR01_WIN2008R2, in the "Name" field and the description "Test Server 01". The user then clicked the add button, which saves the host data to the "hosts" database and displays the host information in a table on the hosts tab as shown in Figure 34.

Figure 34. Hosts Information Table

After the host entry for the test server had been created the delete and update functions were tested. The

user selected the host name "SVR01_WIN2008R2," which displayed the update and delete button as seen in Figure 35.

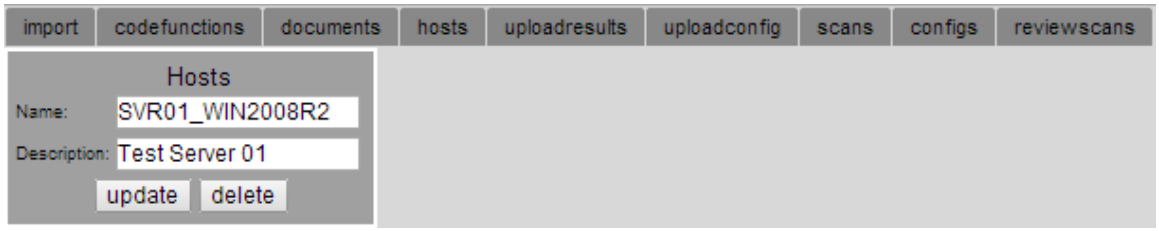


Figure 35. Hosts Update / Delete Dialog

The user clicked the update button and tested modifying the host name and description. The updated host was selected, and the user selected the delete button, which removed the record. The SVR01_WIN2008R2 host was added back and the user proceeded to the uploadresults tab for testing the upload scan results function.

3. Upload SCAP Baseline Scan Results

The user selected the uploadresults tab to bring up the upload results dialog window seen in Figure 36.

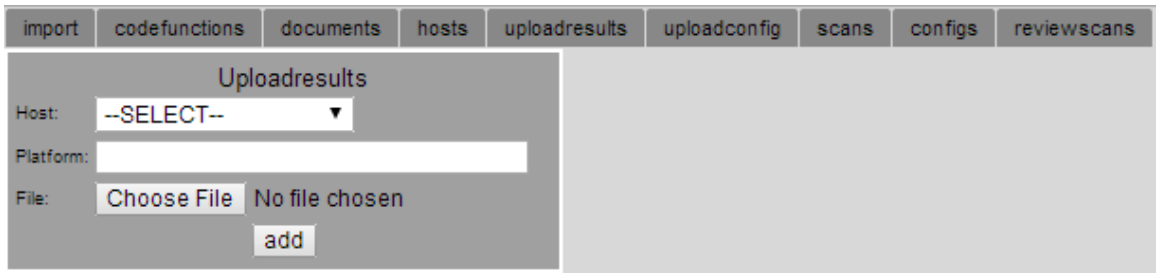


Figure 36. Upload Results Dialog

Using the scan result files and data provided in the "Test Server SCAP Benchmark Result Files" table, the user

added each of the result files for the host SVR01_WIN2008R2 by selecting the host from the dropdown menu and entering the associated platform in the platform field. As the results are uploaded and added to the database, the data for each uploaded result appears in the upload results table under the uploadresults tab, seen in Figure 37.

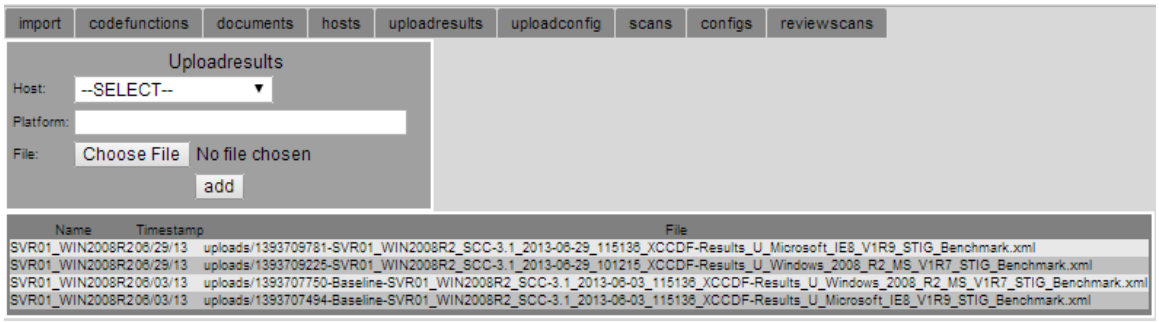


Figure 37. Upload Results Table

After the scan results were uploaded, the user tested the ability to update and delete uploaded scan results. The user selected the host name "SVR01_WIN2008R2," which displayed the update and delete button as seen in Figure 38.

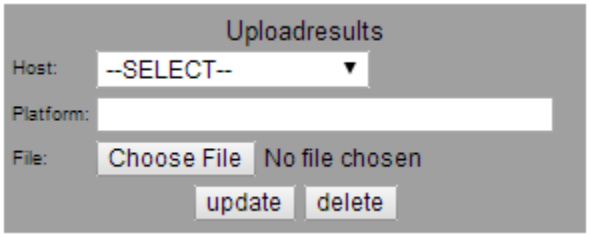


Figure 38. Upload Results Update / Delete Dialog

The user clicked the update button and tested modifying the host, platform, and scan result file. The updated scan result was selected and the user selected the

delete button to remove the record. The deleted scan result file was added back and the user proceeded to the scans tab for testing the view scan results function.

4. View Scan Results

The user selected the scans tab, which displayed the scan results list as showing in Figure 39. This list displays the date, host, platform, and file name for the uploaded scan results.

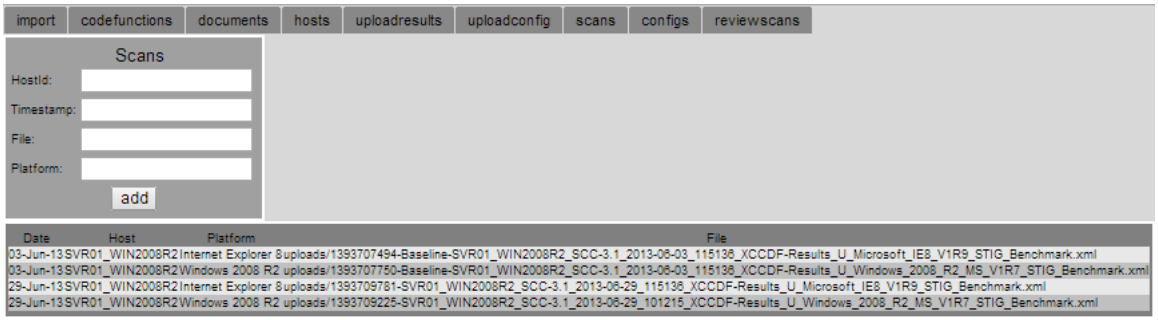


Figure 39. View Scans Table

The user selected the date in the first column of the view scans table, which displayed the data related recorded with the uploaded scan result. Figure 40 displays the scans dialog that permits the user to update data related to each uploaded scan result.

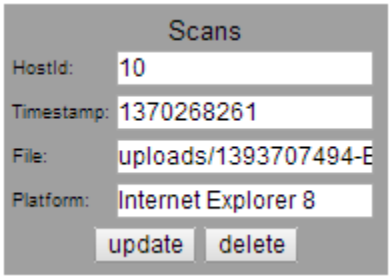


Figure 40. Update / Delete Scans Dialog

The user tested updating the platform associated with a selected scan result. The user clicked the delete button, which removed the modified scan result. The deleted scan result was then uploaded and added from the uploadscans tab.

5. Review Scans

The user selected the reviewscans tab to test the comparative functions for analyzing the baseline scan results with the modified target results. The review scans table is shown in Figure 41.

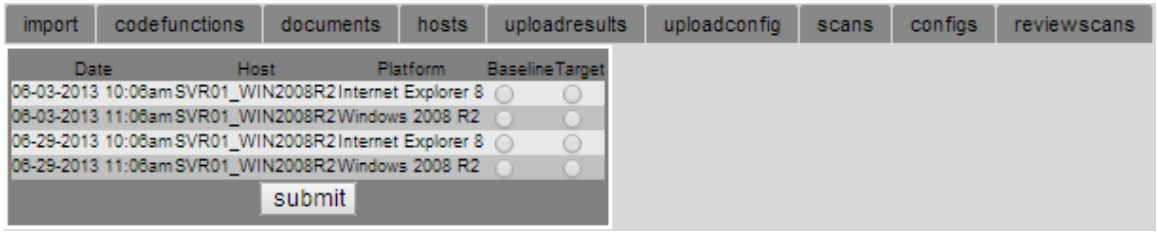


Figure 41. Review Scans Table

The user selected the Internet Explorer 8 scan results from 06-03-2013 as the baseline and selected the 06-29-2013 result as the target. The user selected the submit button, which produced a results comparison table as seen in Figure 42. As expected, the vulnerabilities that had mismatched values between the baseline and target were highlighted in red and matched the configuration changes made for the target result scan.

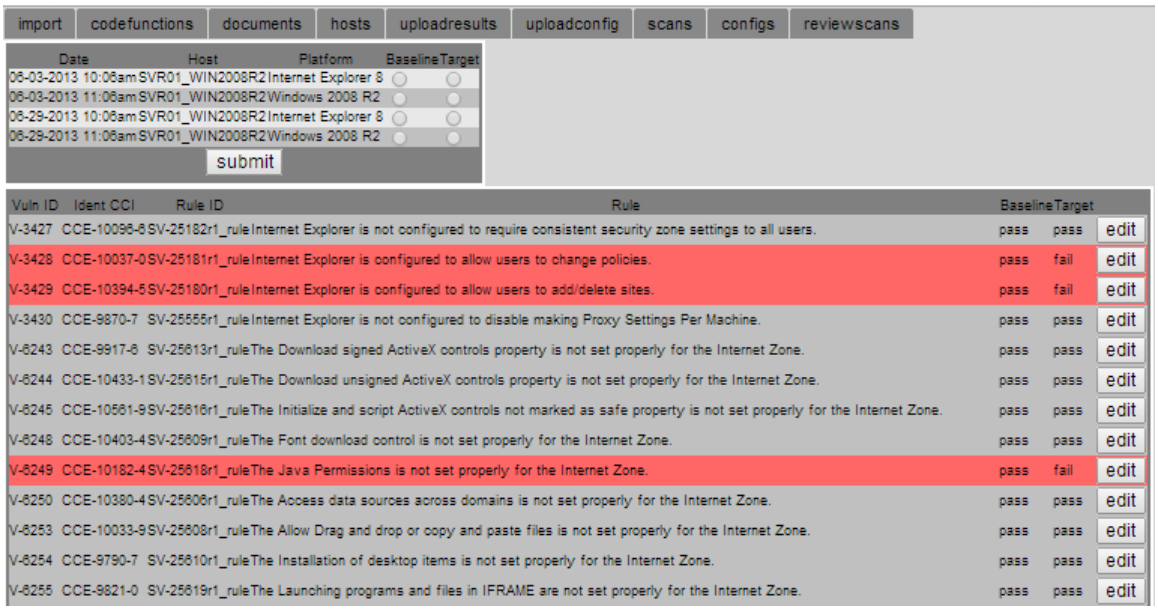


Figure 42. Internet Explorer Scans Comparison

The review scans comparison table displays the Vuln ID, Ident CCI, Rule ID, Rule and the results for the baseline and target files. The Vuln ID represents the unique vulnerability identifier that is used for identifying vulnerabilities in VMS. The Ident CCI column displays the CCE ID used by the NVD for identifying unique system configuration related vulnerabilities. The Rule ID is used within the SCAP XCCDF and benchmark result files to denote a specific automated check. The rule column provides the title or a short description of the vulnerability check.

The user selected the Windows 2008 R2 scan results from 06-03-2013 as the baseline and selected the 06-29-2013 result as the target. The user selected the submit button, which produced a results comparison table as seen in Figure 43. As expected, the vulnerabilities that had mismatched

values between the baseline and target were highlighted in red and matched the configuration changes made for the target result scan.

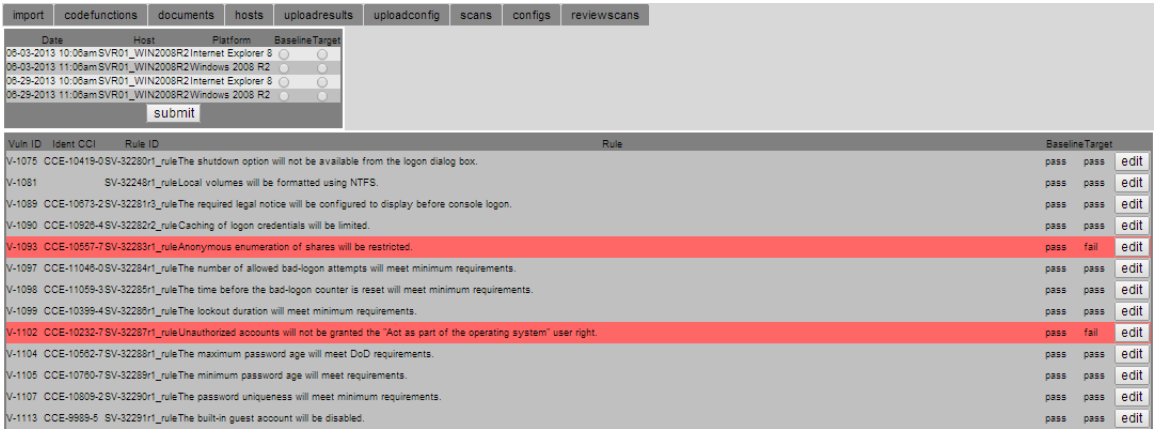


Figure 43. Windows 2008 R2 Scans Comparison

This concluded the functional testing for the server validation components of the proof-of-concept system.

B. NETWORK DEVICE FUNCTIONAL TESTING

In order to validate network device functionality, a pair of configuration files needed to be created. A Cisco 3560E layer-2 switch configuration file was used for network device testing. For the baseline, the configuration setting associated with finding V-3085, titled "The network element must have HTTP service for administrative access disabled," was set to a compliant state. The updated config was set to a non-compliant state to represent a switch that had fallen out of compliance. These configuration files would be used later in the testing.

1. Preparing Custom Checks

The XCCDF XML content file, "U_L2_Switch_Cisco_V8R16_Manual-XCCDF.xml" was imported into the proof-of-concept system using the same procedure described in the previous section. Once the import was completed, the user selected the documents tab and was presented with the document title and xml file names as shown in Figure 44.

Figure 44. Document List

From here, the user clicked the select button for the U_L2_Switch_Cisco_V8R16_Manual-XCCDF.xml file, which produced the table shown in Figure 45. This table includes a selectable list of profiles made up of MAC and sensitivity levels.

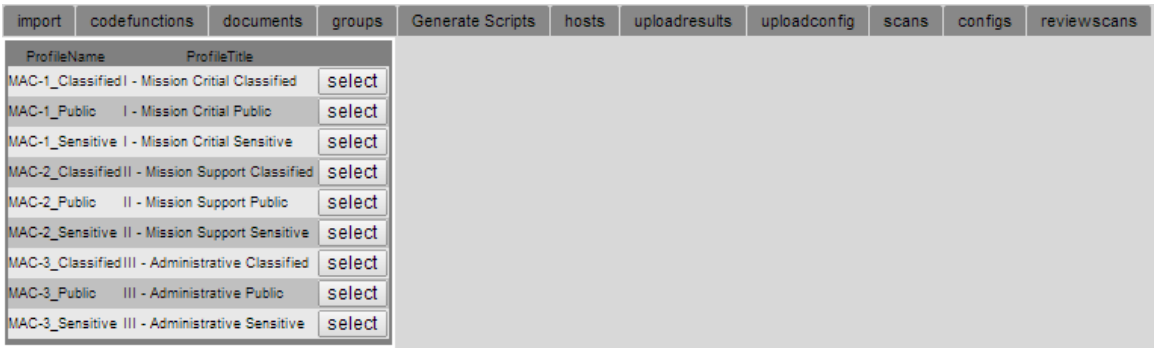
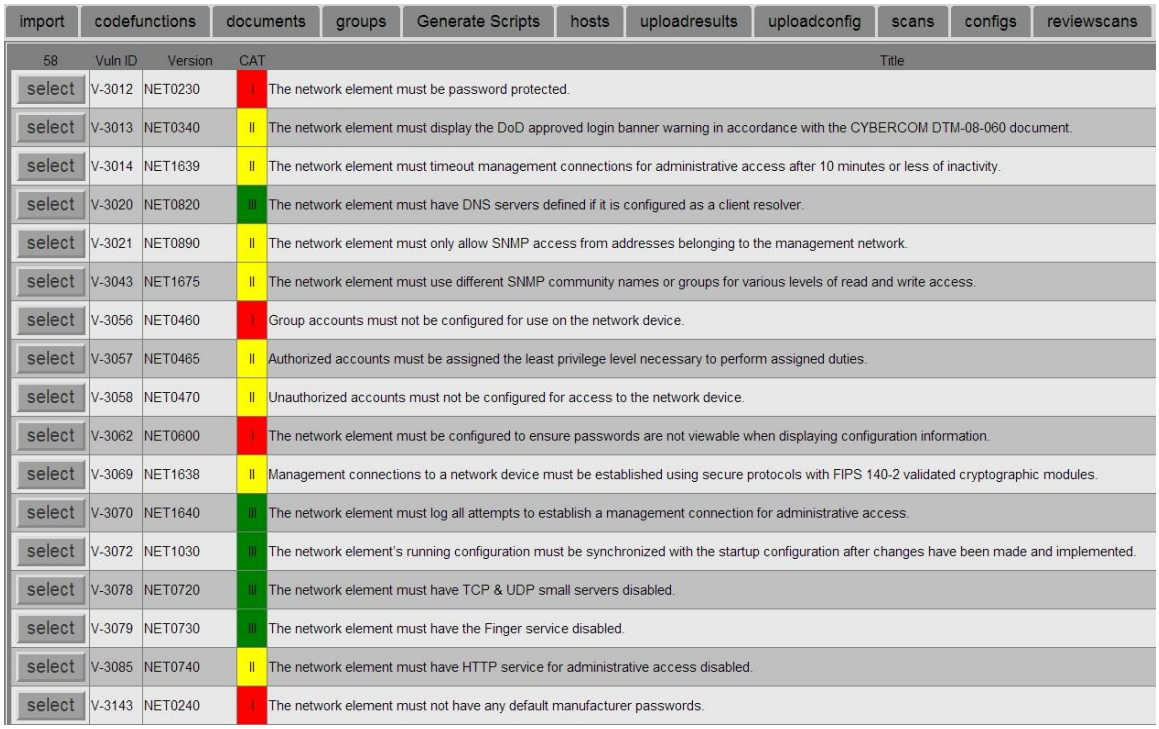


Figure 45. Document Profiles List

The user clicked on the select button next to the "MAC-2_Sensitive II-Mission Support Sensitive," which produced a list of all vulnerabilities associated with that MAC and sensitivity level for the selected XCCDF document. A truncated version of that output is show in Figure 46.



import | codefunctions | documents | groups | Generate Scripts | hosts | uploadresults | uploadconfig | scans | configs | reviewscans

58	Vuln ID	Version	CAT	Title
select	V-3012	NET0230	I	The network element must be password protected.
select	V-3013	NET0340	II	The network element must display the DoD approved login banner warning in accordance with the CYBERCOM DTM-08-060 document.
select	V-3014	NET1639	II	The network element must timeout management connections for administrative access after 10 minutes or less of inactivity.
select	V-3020	NET0820	III	The network element must have DNS servers defined if it is configured as a client resolver.
select	V-3021	NET0890	II	The network element must only allow SNMP access from addresses belonging to the management network.
select	V-3043	NET1675	II	The network element must use different SNMP community names or groups for various levels of read and write access.
select	V-3056	NET0460	I	Group accounts must not be configured for use on the network device.
select	V-3057	NET0465	II	Authorized accounts must be assigned the least privilege level necessary to perform assigned duties.
select	V-3058	NET0470	II	Unauthorized accounts must not be configured for access to the network device.
select	V-3062	NET0600	I	The network element must be configured to ensure passwords are not viewable when displaying configuration information.
select	V-3069	NET1638	II	Management connections to a network device must be established using secure protocols with FIPS 140-2 validated cryptographic modules.
select	V-3070	NET1640	III	The network element must log all attempts to establish a management connection for administrative access.
select	V-3072	NET1030	III	The network element's running configuration must be synchronized with the startup configuration after changes have been made and implemented.
select	V-3078	NET0720	III	The network element must have TCP & UDP small servers disabled.
select	V-3079	NET0730	III	The network element must have the Finger service disabled.
select	V-3085	NET0740	II	The network element must have HTTP service for administrative access disabled.
select	V-3143	NET0240	I	The network element must not have any default manufacturer passwords.

Figure 46. XCCDF Document Vulnerability List

To create a custom check, the user selected vulnerability V-3012, "The network element must be password protected." The user was then redirected to the interface for custom check creation. The user then selected "Cisco Config Null is Bad" from the functions drop down and inserted this previously created code function. This interface, with the inserted code function, appears in Figure 47.

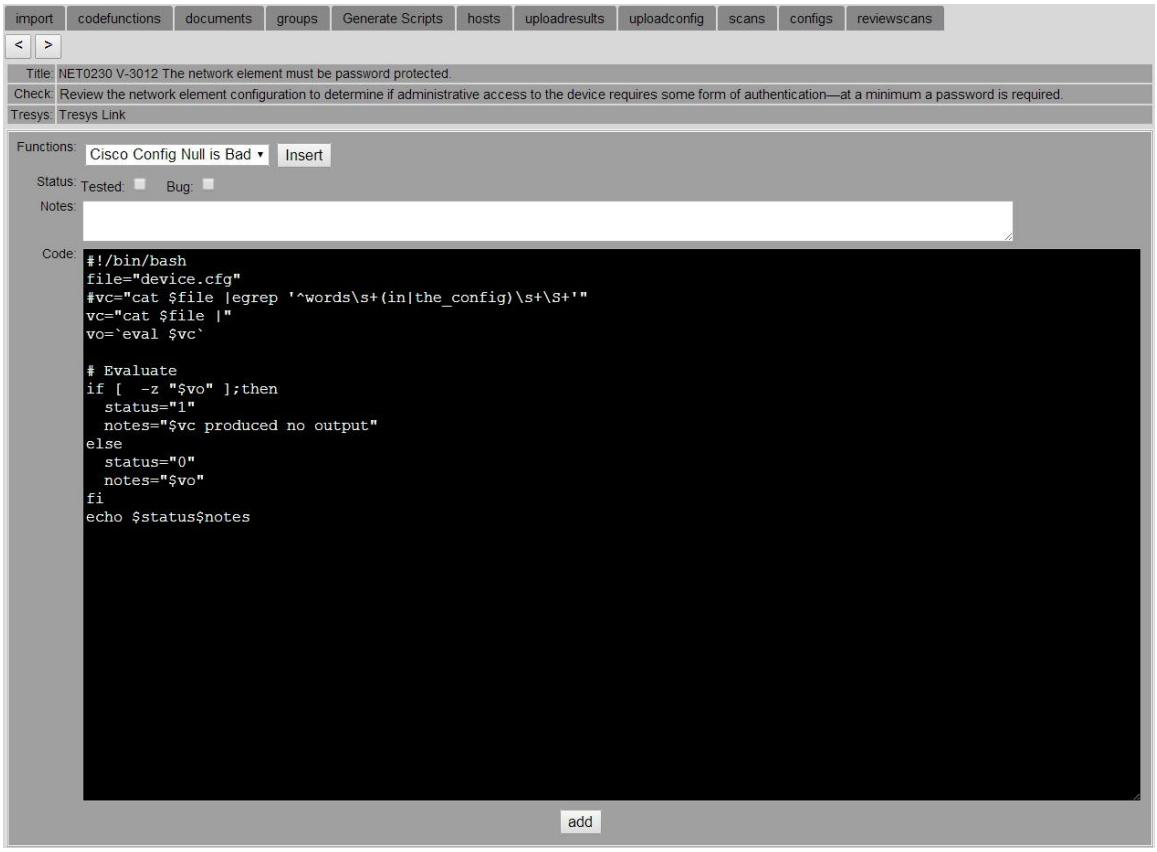


Figure 47. Vulnerability Check Creation

The user then modified the existing evaluation script to check the configuration file for a line beginning with the words “enable secret” or “enable password”. To do this, the line #vc="cat $file |egrep '^words\s+(in|the_config)\s+\S+'" was uncommented, by removing the # mark, and changed to vc=”cat $file |egrep ‘^enable\s+(secret|password)\s+\S+’”. The final state of the check is shown in Figure 48.

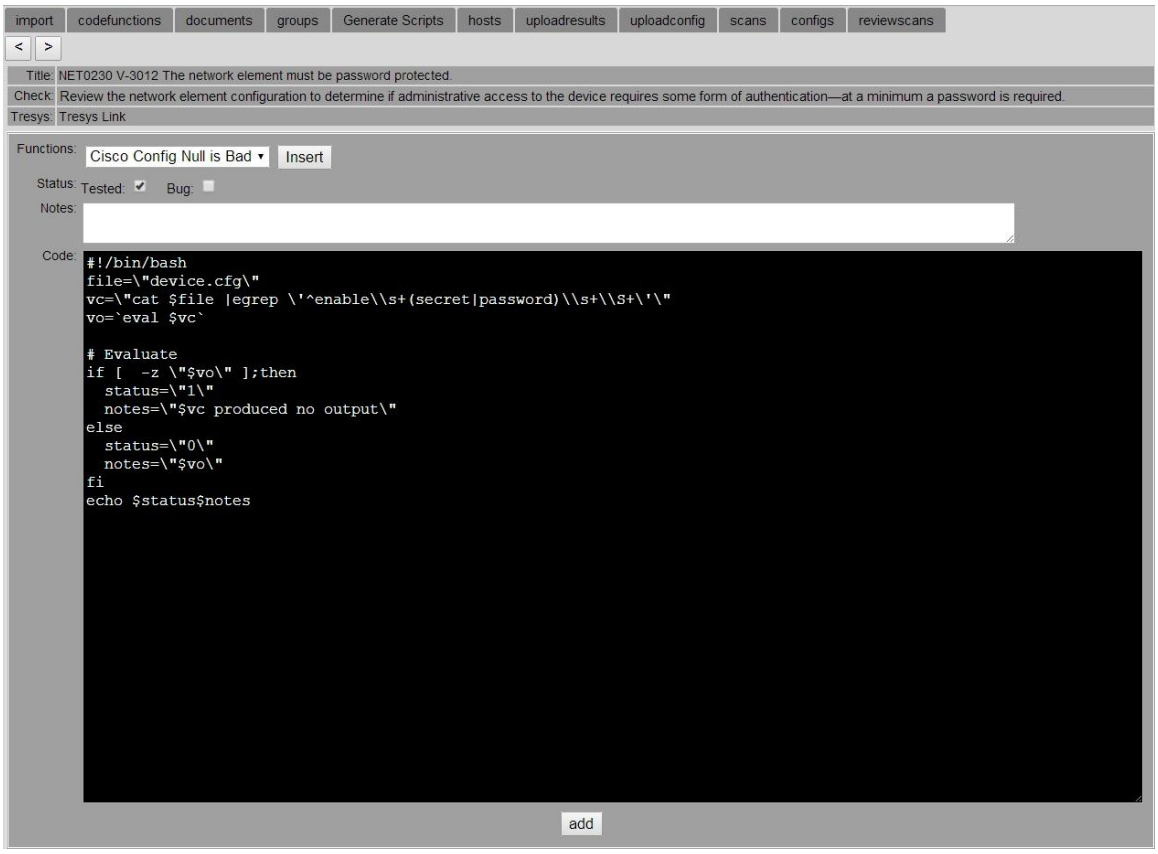


Figure 48. Vulnerability Check

The check was then added by clicking the add button. In normal operation, this new check could be run against a known configuration and validated as good or bad. This particular check was known to be good and was marked as such by the user who checked the "Tested" box. This process was repeated for several other checks. Some of these checks also tested as good, while others did not. Finally some checks were started, but never marked as good or bad. The select button next to these checks of varying status appear in the colors green, red and yellow to signify good, bad and unknown respectively as shown in Figure 49.

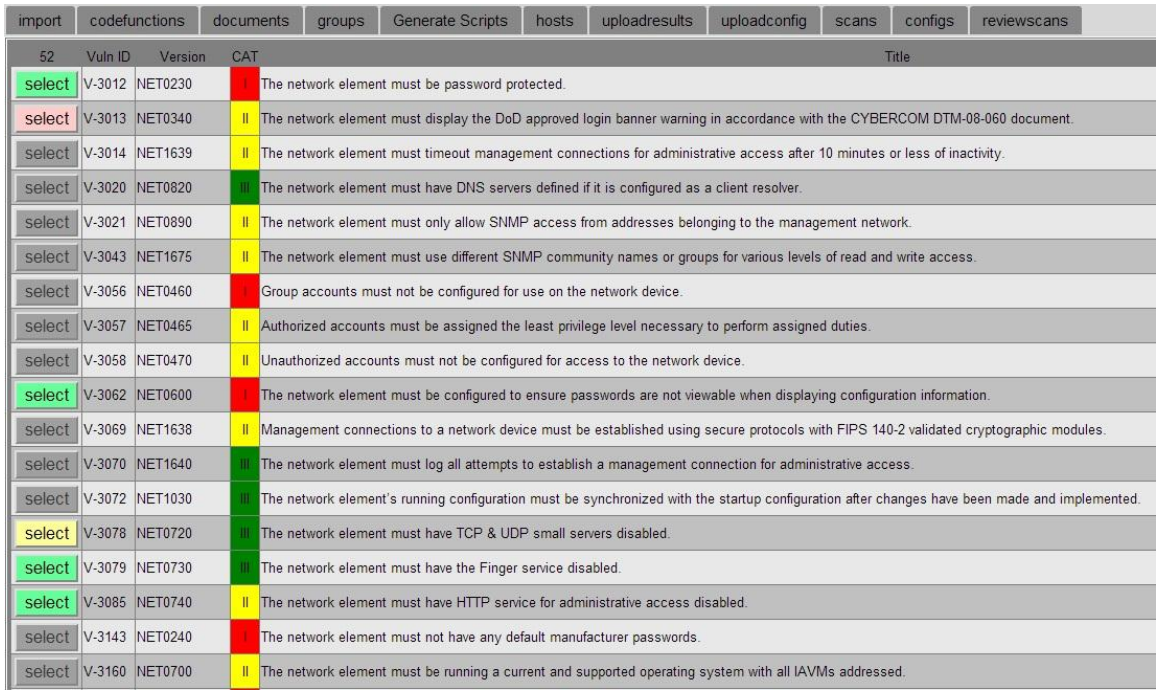


Figure 49. Custom Check Status

2. Validation and Comparison

Once the custom checks were created, it was time to begin the testing of validation and comparison. In order to do this, the network configuration files previously created needed to be uploaded into the system. To do this the user started by clicking on the uploadconfig tab. Doing so displayed the interface shown in Figure 50.

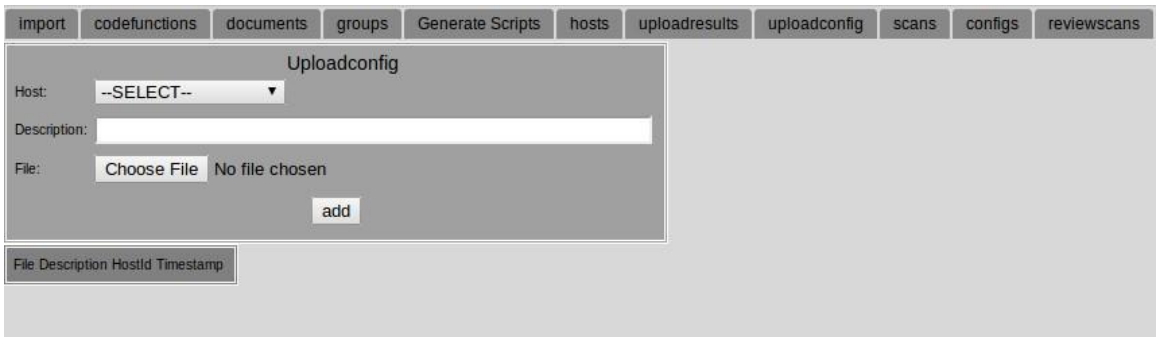


Figure 50. Uploadconfig Dialog

The user then selected the host Tester_3560E and clicked on the "Choose File" button to select a configuration file from his local machine. This dialog is shown in Figure 51.

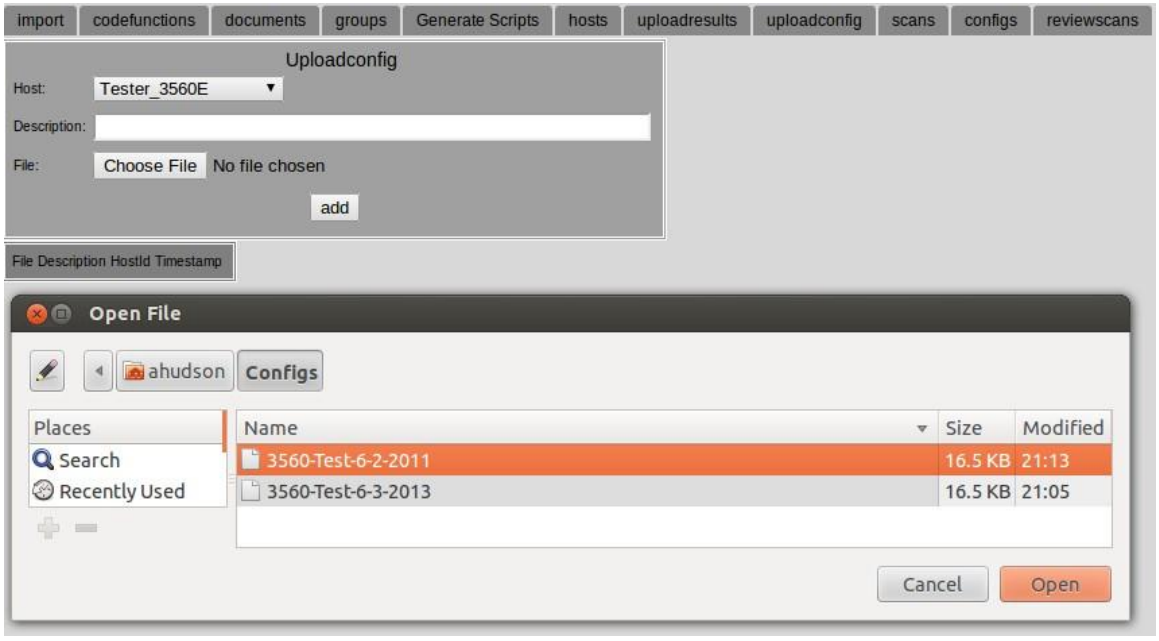


Figure 51. Choosing a File Dialog

The user selected the original file and click open to confirm the selection. The user then entered a relevant description in the description field. This is show in Figure 52.

Figure 52. Uploading a Configuration File

After clicking the add button and running through the procedure a second time to upload the updated configuration, the user was presented with updated page shown in Figure 53.

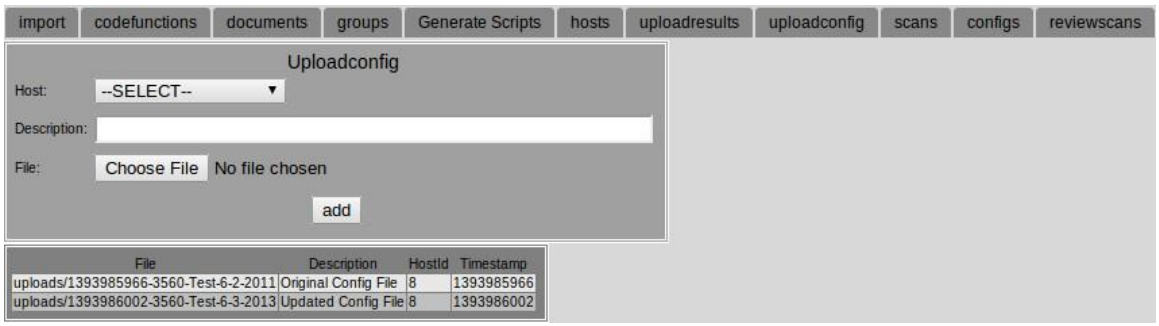


Figure 53. Uploaded Configuration Files

Once the configuration files were uploaded, the user moved on to the configs tab where the configurations could be selected and loaded into a temporary file for validation scanning. These options are shown in Figure 54.

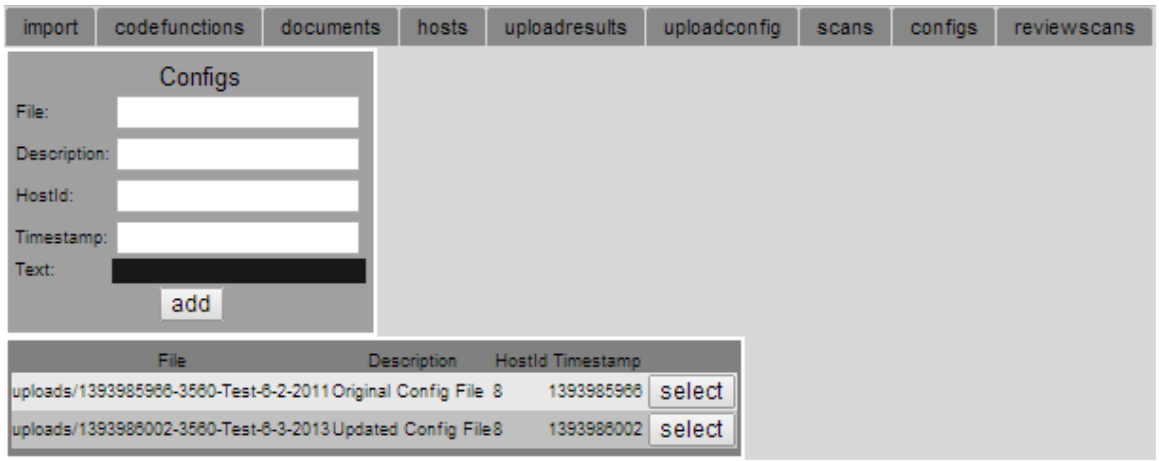


Figure 54. Initial Configs Tab

By clicking on the file path and name, the configuration file was presented in the text box for visual inspection by the user. A truncated version of the text field and the other editable fields are shown in Figure 55.

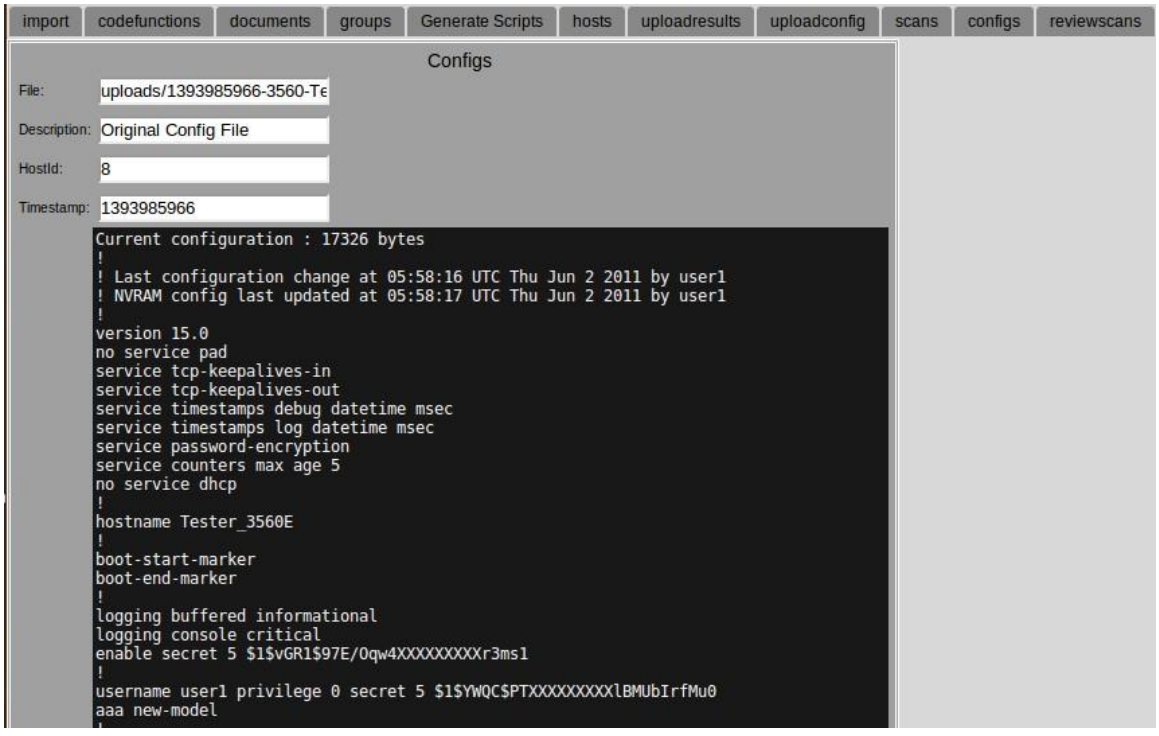


Figure 55. Selected Config File View

From here the user scrolled down to the bottom of the page where he had to option to update the file, description, host id or timestamp field, delete the entire configuration file from the system or select one of the configuration files for validation scanning. This is shown in Figure 56.

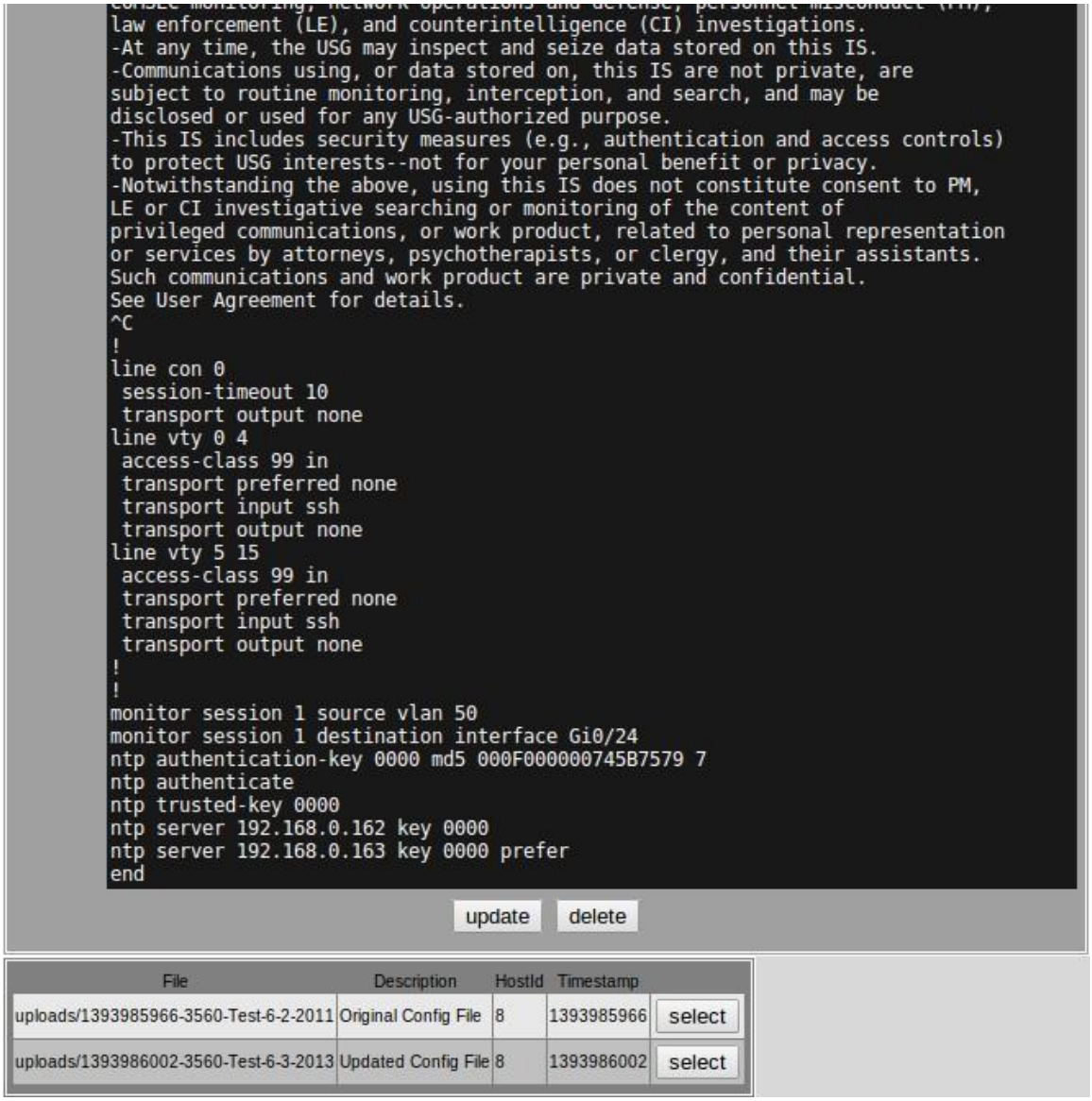


Figure 56. Update, Delete or Select Config Options

For the proof-of-concept testing, the user selected the original configuration file. This is indicated by the words "Config has been selected" as shown in Figure 57.

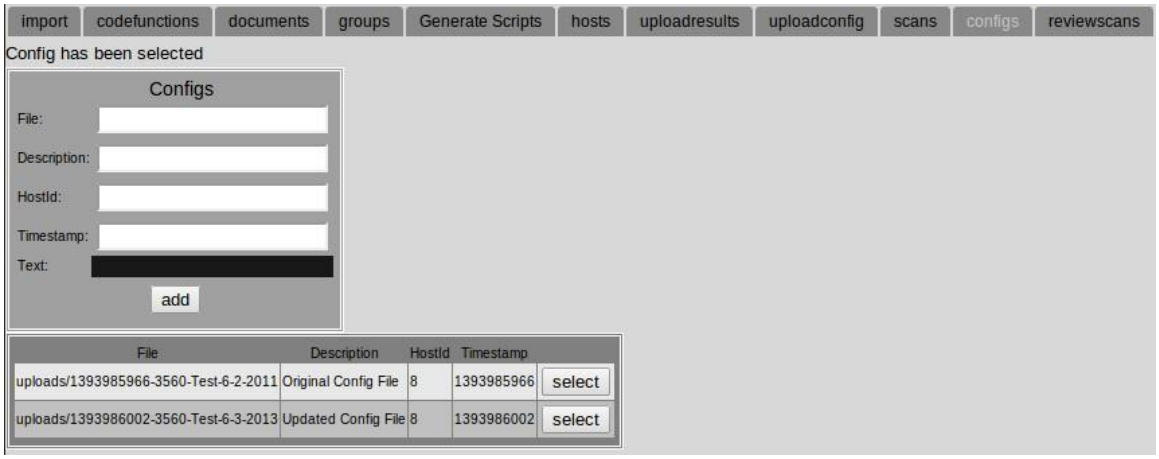


Figure 57. Configuration Selected

Once a configuration file was selected, the user clicked on the generate script tab to generate the scripts for custom checks created earlier in the proof-of-concept testing. Each individual check is converted into a script based on the version name of the check. The user was then presented with the web page in Figure 58.

Figure 58. Scripts Generated

From here the user selected the previously configured switch host and entered information identifying the scan as the original as shown in Figure 59.

Figure 59. Execute scan

The user then selected the scan button to run the validation. The results of the scan were entered into the database as well as being displayed as shown in Figure 60.

Figure 60. Scan of Original Config

The user reviewed the validation results and determined that original configuration file passed three of

the four checks. The only failing check was V-3079, titled "The network element must have the finger service disabled". To verify that the validation results were uploaded the user clicked on the scans tab and was presented with the webpage shown in Figure 61.

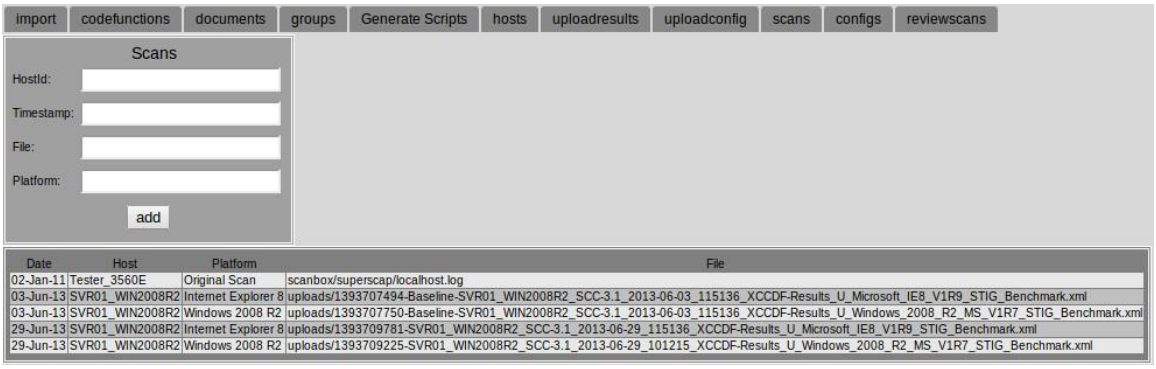


Figure 61. Uploaded Scans

The user then repeated the process for validating the updated configuration file. After the user executed the scan of the updated configuration file, he was presented with the webpage shown in Figure 62.

Figure 62. Scan of Modified Config

The user then verified that the update configuration file failed both V-3079, as the original had, and V-3085 titled "The network element must have HTTP service for administrative access disabled," which the original did not. This was the setting that was toggled on purpose to illicit a difference in validation reports between the original and updated configuration files. From here the user clicked on the review scans tab and was presented with the webpage shown in Figure 63. The user then selected the original validation as the baseline and the updated validation as the target.

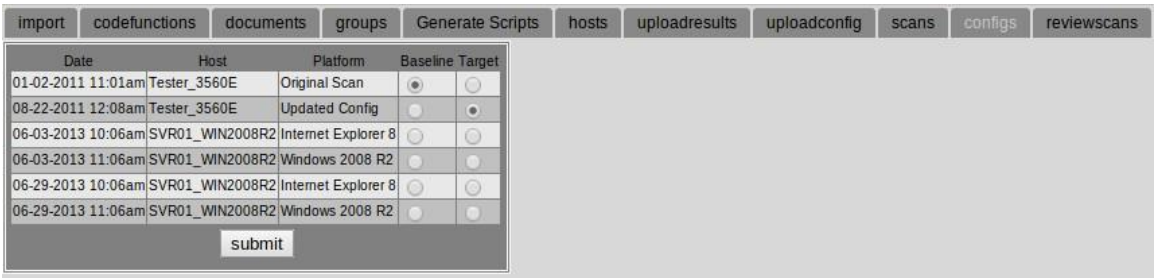


Figure 63. Review Scans for Network Device

The user then clicked the submit button to compare the two results. As expected, the proof-of-concept system identified V-3085 as differing between the baseline and the target as show in Figure 64.

Figure 64. Network Results Comparison

This completed the testing for network validation component. This concluded the proof-of-concept system functional test.

VI. CONCLUSION

A. PROOF-OF-CONCEPT SYSTEM RESULTS

The proof-of-concept system testing demonstrates capabilities that address several areas of need that the current DoD-mandated tools do not. The ability to digest, archive, and compare both SCAP and custom-written security validation results for individual assets or asset types proves valuable in several use case scenarios.

In isolated development environments where security settings may be adjusted as part of application testing, an organization may not want, or be able to, use CMRS. An organization's requirement to isolate their development environment may preclude them from utilizing a solution that must have external connectivity in order to report or update security content.

CMRS reporting, in its current preliminary state, does not support reporting risk scores associated with individual assets, instead providing an overall risk score for all monitored organizational assets. A CMRS score associated with an organization's assets is a raw score, which cannot be altered from the compliance data provided by reporting agents, such as HBSS or ACAS. The initial deployment phase of CMRS does not support the modification of scoring due to risk mitigation or the identification of false positives. The insertion of POA&Ms for specific reported findings is also currently unsupported.

The proof-of-concept system has shown the ability to address both of these issues. This system does not require external connectivity for updates of security content or

reporting, so it can operate in a standalone or "air-gap" network. The proof-of-concept system also allows for a single asset to be compared to other assets of the same type or for a previous version of that asset to be compared to a later version of that same asset. In essence, whether operating in a closed or connected environment the proof-of-concept system allows users to identify a standard set of security settings that make up a system security baseline and compare those results to results generated against the same system or same type of system at a later date in development. These capabilities prove especially valuable when conducting engineering and development activities.

Another advantage of the proof-of-concept system is the ability to support custom written validation checks. This allows the system to validate network device configurations against specific security checks, which is especially useful when SCAP content for a device does not exist. This capability is also useful when scripting or staging network configurations since network engineers often pre-build or script a configuration prior to loading it on a device. This saves time during an install and allows others to review their work prior to deployment. The system's ability to parse through flat files searching for user specified security settings makes it ideal for these purposes.

The proof-of-concept system has shown that it is capable of meeting some immediate needs for both servers and network devices, but there are some short-comings. In its present form, the system does not utilize an

authentication mechanism for restricting access. The overall flow of the system could be improved as it relates to network validation actions and the process of writing custom checks requires a fairly strong understanding of shell scripting in order to parse and identify target data. These shortcomings and a number of potential improvements are the focus of the next section.

B. IMPROVEMENTS

The following sections address improvements that could be made to the system to further enhance its capability.

1. Role Based Access Control

The proof-of-concept system can be deployed to a single user's laptop, workstation, or virtual machine. In these instances, access control to individually assigned assets is often controlled via corporate security policies. These restrictions are implemented to prevent external users from accessing content on another individual's asset. However, when the proof-of-concept system is deployed in a shared environment where multiple users may utilize it, access control needs to be established.

When multiple users utilize the system, data detailing the security posture of IT from various parts of the organization may be present on the same system. In this case, controls must be in place to control the confidentiality and integrity of the user's data. RBAC is an ideal approach because it allows personnel from different areas of an organization to be assigned various roles. In the case of the proof-of-concept system, these roles could be defined in many different ways. For example,

these roles could be based on what a user should be able to see, change, delete or create. When implemented, the specific requirements of an organization would define exactly what roles were created.

2. System Flow

As detailed in Chapters IV and V, the system utilizes tabs that represent each specific functional requirement of the proof-of-concept system. These tabs define the layout of the GUI. The GUI could be improved to provide a more intuitive interface that more clearly represents the process for evaluating an asset.

For example, the system layout could be broken into network device and server sections. This would eliminate tabs that were not relevant to a particular section, cleaning up the overall appearance of the GUI. Another example might be to allow users to select and run scans directly from the configs tab or for user to have the option to upload XCCDF results files while simultaneously defining a new host.

While the system is intentionally designed in this tabbed format to showcase each individual function independently, it could be modified to provide a more user-friendly operating environment.

3. Custom Checks

Custom checks provide a framework for vulnerability assessment as it relates to network devices. In the case of the proof-of-concept system the checks associated with each vulnerability ID are written from scratch in shell scripting or created by modifying code from a selected

template. In either case, even with the ability to check scripts from the command line, a moderate understanding of programming is needed to ensure that the information being searched is identified when present and is identified as missing when it is not present. In the Figure 65, a check has been written using the "Cisco Null is Bad" template. The vulnerability being evaluated is meant to ensure that administrative access to the network device is password protected.

Code:

```
#!/bin/bash
file="device.cfg"
vc="cat $file |egrep '^enable\s+(secret|password)\s+\S+'"
vo=`eval $vc`

# Evaluate
if [  -z "$vo" ];then
  status="1"
  notes="$vc produced no output"
else
  status="0"
  notes="$vo"
fi
echo $status$notes
```

Figure 65. Password Custom Check

A validator inspecting this vulnerability on a Cisco switch or router needs to ensure that "enable secret" or "enable password" is present in the device's configuration file. The script above uses both the cat and egrep commands. The cat command is usually used to display a file. The egrep command is usually used to search text for a specific set of characters. When they are combined in the manner above, the device.cfg file is parsed looking for a line beginning with the word "enable" followed by one or more spaces and then either the word "secret" or the word

"password". Writing these checks becomes more challenging as the acceptable set of strings grows and dependencies become relevant.

To simplify and standardize the way network checks are created, the code used could be derived from an improved set of templates. These templates would allow users to enter commands or attributes of interest into various fields associated with regular expressions such as "and," "not," "matches" and "contains," and the checks would automatically be created. Based on the fields used, the check would search for the presence or the absence of specific text to validate a check. By standardizing the way checks are created, network validation results would be more consistent and easier to create.

C. FUTURE WORK

There are several areas where future work should be focused. It would be worthwhile to expand the capabilities of the network validation functionality. As described in the previous section, a more user-friendly template for creating checks would be particularly useful in this regard. DISA-provided network-checklists come in generic roles or functions, and device specific varieties.

For example, in the proof-of-concept system both the Layer 2 Switch Security Technical Implementation Guide—Cisco and the Layer 2 Switch Security Technical Implementation Guide—Generic were loaded. The Cisco version of the guide has Cisco IOS specific checks for the various vulnerabilities identified. The generic version of the guide does not apply to a specific vendor product or operating system. This implies that the same vulnerability,

referenced in each guide, may require multiple checks to be written. Each version of the check would then need to be assigned to a specific vendor, operating system, or even a specific model of device. This would all need to be tracked within the database and the process for creating the scripts, running them and uploading the results into the database would need to be modified.

Another way to expand the capabilities of the proof-of-concept system would be to address continuous monitoring. There are several approaches that could be taken here, but the most straightforward would take advantage of most OS' abilities to schedule jobs and utilize network files systems. On systems where continuous monitoring is desired, administrators could schedule existing SCAP compliant tools to run validation scans and save the results to a network file-share. The proof-of-concept tool could monitor these various file-shares while consuming and cataloging the results as they appeared. Significant changes to the proof-of-concept system would be required to add this automation feature. The system would need some way of knowing which new result files belong to which systems, though it is possible that this information could be pulled from the XCCDF benchmark results files. Ideally, the system would have the ability to compare the most recent results against the baseline for a given system and notify users of any changes that affected the risk assessment of an asset.

The same challenges would exist for network device validation, but the process would be a little different. In this scenario it would make more sense to automate the

process of attaining device configuration files. With specific settings needed for each device, the cataloguing or assignment of a specific device configuration, as well as the validation results, would most likely be tied to the initial process of downloading that specific device's configuration file. In this way, the proof-of-concept tool would know which device and validation checks to be run before it even attempted to retrieve a device configuration. Having the ability to provide a near real-time status on the network devices and servers within a particular environment would provide an organization with valuable information on the security posture of the monitored assets in their environment.

To evaluate the usability of the system, several potential studies could be conducted. The proof-of-concept system could be provided to assessors for use in a real world evaluations or compliance monitoring scenarios. It could also be piloted or tested in a scenario, where some assessors would have access to the proof-of-concept system and others would not, that could illustrate the effect on time savings and accuracy. Finally, the proof-of-concept system could be used in a classroom system to explore the compliance process and maintenance through an example. Each of these scenarios would provide valuable feedback that could shape future versions of the tool and provide an enhanced understanding of its usability.

These improvements and suggestions for future work aim to address shortcomings and extended capabilities that are needed to integrate the proof-of-concept system into a production security monitoring system capable of providing

automated compliance validation and continuous monitoring. An open source system like this could be tailored and enhanced to meet the specific needs of individuals and organizations to provide security monitoring and/or augment their existing tool sets.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX A. PROOF-OF-CONCEPT DATABASE STRUCTURE

For the proof-of-concept, a database named SCANS was created. This database contains many tables. Some of the tables were created manually, while others were created as part of script execution. A short description of each table, along with basic characteristics of each field within each table can be found below.

A. CODE

Description: This table stores the custom code created to assess the status of each vulnerability.

Column	Type	Null
id	int(11)	No
groupId	int(11)	Yes
creatorId	int(11)	Yes
fnId	int(11)	Yes
code	text	Yes
notes	text	Yes
selected	varchar(3)	Yes
codeTypeId	int(11)	Yes
tested	varchar(2)	Yes
bug	varchar(2)	Yes

Table 7. Code Table Data Columns

B. CODEFUNCTIONS

Description: This table stores the code functions that can be used as a template for custom code.

Column	Type	Null
id	int(11)	No
name	varchar(100)	Yes
description	text	Yes
code	text	Yes
codeTypeId	int(11)	Yes
tested	int(11)	Yes
creatorId	int(11)	Yes
variables	varchar(200)	Yes
execute	varchar(2)	Yes

Table 8. Codefunctions Table Data Columns

C. CONFIGS

Description: This table stores the uploaded device configuration files that can be validated by the proof-of-concept system.

Column	Type	Null
id	int(11)	No
file	varchar(200)	Yes
description	varchar(50)	Yes
hostId	int(11)	Yes
timestamp	int(11)	Yes

Table 9. Config Table Data Columns

D. DOCUMENTS

Description: This table stores all the information parsed from the XCCDF XML documents.

Column	Type	Null
id	int(11)	No
documentTitle	varchar(200)	Yes
documentDescription	text	Yes
documentPublisher	varchar(50)	Yes
documentSource	varchar(50)	Yes
documentHref	varchar(50)	Yes
documentRelease	varchar(100)	Yes
documentReleaseVersion	varchar(50)	Yes
xmlNsDsig	varchar(100)	Yes
xmlNsXhtml	varchar(100)	Yes
xmlNsXsi	varchar(100)	Yes
xmlNsCpe	varchar(100)	Yes
xmlNsDc	varchar(100)	Yes
xmlId	varchar(100)	Yes
xmlLang	varchar(100)	Yes
xmlSchemaLocation	varchar(200)	Yes
xmlNs	varchar(100)	Yes
documentDate	varchar(100)	Yes
xsiSchemaLocation	varchar(200)	Yes
xmlfile	varchar(200)	Yes

Table 10. Documents Table Data Columns

E. GROUPS

Description: This table stores information about each requirement described in the XCCDF XML documents.

Column	Type	Null
id	int(11)	No
vulnId	varchar(20)	Yes
ruleId	varchar(20)	Yes
severity	varchar(20)	Yes
weight	varchar(20)	Yes
version	varchar(50)	Yes
title	text	Yes

Column	Type	Null
description	text	Yes
falsePositives	varchar(200)	Yes
falseNegatives	varchar(200)	Yes
documentable	varchar(200)	Yes
mitigations	varchar(200)	Yes
severityOverrideGuidance	varchar(200)	Yes
potentialImpacts	varchar(200)	Yes
thirdPartyTools	varchar(200)	Yes
mitigationControl	varchar(200)	Yes
responsibility	varchar(200)	Yes
iaControls	varchar(200)	Yes
dcTitle	varchar(200)	Yes
dcPublisher	varchar(50)	Yes
dcType	varchar(50)	Yes
dcSubject	varchar(50)	Yes
dcIdentifier	varchar(50)	Yes
identSystemUrl	varchar(100)	Yes
identCci	varchar(100)	Yes
fixRefId	varchar(100)	Yes
fixText	text	Yes
fixId	varchar(100)	Yes
chkId	varchar(100)	Yes
checkContentRef	varchar(50)	Yes
checkContentHref	varchar(100)	Yes
checkText	text	Yes
fnId	int(11)	Yes
noFn	int(1)	Yes
referenceId	varchar(20)	Yes

Table 11. Groups Table Data Columns

F. HOSTS

Description: This table stores the list of hosts. It is used as a data source to associate a particular host with each uploaded scan results document.

Column	Type	Null
id	int(11)	No
name	varchar(50)	Yes
description	varchar(100)	Yes

Table 12. Hosts Table Data Columns

G. PROFILES

Description: This table stores the various profiles (e.g., MAC-1 Classified, MAC-2 Public) contained within each uploaded XCCDF file.

Column	Type	Null
id	int(11)	No
documentId	int(11)	Yes
profileName	varchar(100)	Yes
profileTitle	varchar(100)	Yes

Table 13. Profiles Table Data Columns

H. PROFILESMAP

Description: This table stores information relating a profile with its individual group entries (vulnerabilities).

Column	Type	Null
id	int(11)	No
profileId	int(11)	Yes
vulnId	varchar(50)	Yes

Table 14. ProfilesMap Table Data Columns

I. RESULTS

Description: This table stores the scan results.

Column	Type	Null
id	int(11)	No
timestamp	int(11)	Yes
ruleId	varchar(50)	Yes
result	varchar(10)	Yes
identCci	varchar(50)	Yes
scanId	int(11)	Yes
note	varchar(255)	Yes
output	varchar(255)	Yes
status	int(1)	Yes

Table 15. Results Table Data Columns

J. SCANS

Description: This table stores all the information about a particular scan.

Column	Type	Null
id	int(11)	No
hostId	int(11)	Yes
timestamp	int(11)	Yes
file	varchar(255)	Yes
platform	varchar(255)	Yes

Table 16. Scans Table Data Columns

APPENDIX B. PROOF-OF-CONCEPT SOURCE CODE

The php source code for each page of the proof-of-concept application:

A. INDEX.PHP

```
<?php
include "includes.php";
?>
Includes.php
<?php
include "variables.php";
include "functions.php";
include "htmlhead.php";
include "menu.php";
?>
```

B. VARIABLES.PHP

```
<?php

//*** General Variables ***
session_start();
$phpSelf=basename($_SERVER['PHP_SELF']);
$websiteName="SuperSCAP";
date_default_timezone_set('America/New_York');
$now=time();

//*** Framework Database ***
$dbUser="dbuser";
$dbServer="localhost";
$dbPass="dbpassword";
$dbName="scans";
$mysqli = new mysqli($dbServer,$dbUser,$dbPass,$dbName);

//*** Colors ***
$defaultBgColor="d8d8d8";
$defaultFontFace="arial";
$defaultFontSize="10px";
$myRed="af1d0e";
$myBlue="1c5f92";
$myGreen="6d722d";
$myYellow="d4961b";
```

```
$rc1="#c0c0c0"; // list row color 1
$rc2="#e8e8e8"; // list row color 2
$cc=0;
$tc=$rc1;

//*** Reference Variables ***
$page=basename(substr($phpSelf, 0, -4));
if(isset($_POST['setDocumentId'])){
 $documentId=$_POST['setDocumentId'];
 $_SESSION['documentId']=$_POST['setDocumentId'];
}
if((!isset($documentId))&&(isset($_SESSION['documentId'])))
{
 $documentId=$_SESSION['documentId'];
}
if(isset($_POST['setProfileId'])){
 $profileId=$_POST['setProfileId'];
 $_SESSION['profileId']=$_POST['setProfileId'];
}
if((!isset($profileId))&&(isset($_SESSION['profileId']))){
 $profileId=$_SESSION['profileId'];
}
if(!isset($id)){
 $id='';
}
if(isset($_POST['mode'])){
 $mode=$_POST['mode'];
}elseif(isset($_GET['mode'])){
 $mode=$_GET['mode'];
}else{
 $mode="none";
}
?>
```

C. FUNCTIONS.PHP

```
<?php
function getFields($dbTable){
 global $mysqli,$dbName;
 $vars=array();
 $sql="select column_name from information_schema.columns
where table_schema='$dbName' and table_name='$dbTable'
order by ordinal_position";
 $result = $mysqli->query($sql);
  while ($row = $result->fetch_assoc()){
   $var=$row['column_name'];
```

```
   array_push($vars,"$var");
  }
  return $vars;
}

function showFields($dbTable){
global $mysqli,$dbName;
 $vars=array();
 $sql="select column_name from information_schema.columns
where table_schema='$dbName' and table_name='$dbTable'
order by ordinal_position";
 $result = $mysqli->query($sql);
  while ($row = $result->fetch_assoc()){
   $var=$row['column_name'];
   array_push($vars,"$var");
  }
 print "dbName: $dbName<br>";
 print "dbTable: $dbTable<br>";
 foreach($vars as $var){
  print "var: $var<br>";
 }
}
?>
```

D. HTMLHEAD.PHP

```
<html><head>
<title><?php print "$websiteName"; ?></title>
<link rel="shortcut icon" href="images/favicon.ico"
type="image/x-icon" />
<?php
include "css.php";
?>
</head>
<body topmargin=0 leftmargin=0 bgcolor=<?php print
"$defaultBgColor"; ?>>
css.php
<style type=text/css>
a:link { color: black; text-decoration: none }
a:active { color: yellow; text-decoration: none }
a:visited { color: black; text-decoration: none }
a:hover {
color: #c6c6c6;
text-decoration: none
}
```

```
h1{
font-size: 10px;
font-family: serif;
font-style: normal;
}

h2 {
 font: bold 330%/100% "Lucida Grande";
 position: relative;
 color: #464646;
 margin-bottom:0;
 font-size:12px;
}

h2 span {
 background: url(images/gradient-white.png) repeat-x;
 position: absolute;
 display: block;
 width: 100%;
 height: 22px;
}

h4{
font-size: 16px;
font-family: serif;
font-style: normal;
}

td{
 font-family: <?php print "$defaultFontFace"; ?>;
 font-size: <?php print "$defaultFontSize"; ?>;
}

td.menuSpace{
 padding: 0;
}

td.menu{
 font-family: arial;
 font-size: 12px;
 padding: 4 10 4 10;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #888888;
 font-weight: normal;
 -webkit-border-radius: 3 3 0 0 ;
```

```
 -moz-border-radius: 3 3 0 0 ;
 border-radius: 3 3 0 0;
}

td.subMenu{
 font-family: arial;
 font-size: 12px;
 padding: 4 10 4 10;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #888888;
 font-weight: normal;
 -webkit-border-radius: 3 3 0 0;
 -moz-border-radius: 3 3 0 0;
 border-radius: 3 3 0 0;
}

td.menuSel{
 font-family: arial;
 font-size: 12px;
 padding: 4 10 4 10;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #d4961b;
 color: #ebebeb;
 font-weight: bold;
 -webkit-border-radius: 3 3 0 0;
 -moz-border-radius: 3 3 0 0;
 border-radius: 3 3 0 0;
}

td.subMenuSel{
 font-family: arial;
 font-size: 12px;
 padding: 4 10 4 10;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #d4961b;
 color: #ebebeb;
 font-weight: bold;
 -webkit-border-radius: 3 3 0 0;
 -moz-border-radius: 3 3 0 0;
 border-radius: 3 3 0 0;
}

table.form{
```

```
 border-color: #ffffff;
 border-width: 3px ;
 border-style: double;
 border-spacing: 0px;
 padding: 5 5 5 5;
 background-color: #a0a0a0;
}

table.form2{
 border-color: #ffffff;
 border-width: 3px ;
 border-style: double;
 border-spacing: 2px;
 padding: 5px;
 background-color: #a0a0a0;
}

td.formLabel{
 font-family: arial;
 font-size: 11px;
 padding: 2;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #a0a0a0;
 font-weight: normal;
 vertical-align: top;
 text-align: right;
 white-space: nowrap;
}

td.formFieldSmall{
 font-family: arial;
 font-size: 11px;
 padding: 2;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #a0a0a0;
 font-weight: normal;
}

#tooltip1 { position: relative; }
#tooltip1 a span { display: none; color: #black; }
#tooltip1 a:hover span {
 display: block;
 position: absolute;
 background-color: #ffffcc;
```

```
 color: #black;
 padding: 5px;
 border-color: #606060;
 border-style: solid;
 border-width: 2;
 -webkit-border-radius: 6px;
 -moz-border-radius: 6px;
 border-radius: 6px;
 }

#tooltip2 { position: relative; }
#tooltip2 a span { display: none; color: #000000; }
#tooltip2 a:hover span {
 left: 50px;
 display: block;
 position: absolute;
 background-color: #ffffcc;
 color: #000000;
 font-size:14px;
 padding: 5px;
 border-color: #606060;
 border-style: solid;
 border-width: 2;
 -webkit-border-radius: 6px;
 -moz-border-radius: 6px;
 border-radius: 6px;
 }

td.formTitle{
 font-family: arial;
 font-size: 14px;
 padding: 0 0 0 10;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #a0a0a0;
 font-weight: normal;
 text-align: center;
}

td.formSection{
 font-family: arial;
 font-size: 13px;
 padding: 0 0 0 5;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #a0a0a0;
```

```
 font-weight: normal;
 text-align: left;
}


td.formField{
 font-family: arial;
 font-size: 12px;
 padding: 2;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #a0a0a0;
 font-weight: normal;
}

td.formCode{
 font-family: arial;
 font-size: 12px;
 padding: 0;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #000000;
 font-weight: normal;
}

td.formText{
 font-family: arial;
 font-size: 12px;
 padding: 2;
 border-color: #ffffff;
 border-width: 1px;
 background-color: #d0d0d0;
 font-weight: normal;
}

td.formFooter{
 text-align: center;
}

table.list{
 border-color: #ffffff;
 border-width: 3px ;
 border-style: double;
 border-spacing: 1px;
 padding: 5 3 5 3;
 background-color: #a0a0a0;
```

```
 background-color: #808080;
}

td.listTitle{
 text-align: center;
 font-family: arial;
 font-size: 14px;
}

td.listHeader{
 text-align: center;
 font-family: arial;
 font-size: 10px;
}

td.list2{
 font-family: arial;
 font-size: 9px;
 font-weight: normal;
 padding: 2 5 2 5 ;
 border-width: 0;
}

td.list3{
 font-family: arial;
 font-size: 6px;
 font-weight: normal;
 padding: 2 5 2 5 ;
 border-width: 0;
}

.smallText{
font-size:10px;
height: 16px;
}

.smallText2{
font-size:10px;
font-family: arial;
}

.textR{
height: 18px;
}

body {
```

```
     font: 0.8em/21px arial,sans-serif;
}

.checkbox, .radio {
 width: 19px;
 height: 25px;
 padding: 0 5px 0 0;
 background: url(checkbox.png) no-repeat;
 display: block;
 clear: left;
 float: left;
}

.radio {
 background: url(radio.png) no-repeat;
}

.select {
 position: absolute;
 width: 158px;
 height: 21px;
 padding: 0 24px 0 8px;
 color: #fff;
 background: url(select.png) no-repeat;
 overflow: hidden;
 font: 12px/21px arial,sans-serif;
}

.greybutton
{
background-color: #a0a0a0;
color: #383838;
}

.yellowbutton
{
background-color: #ffff99;
}

.greenbutton
{
background-color: #66ff99;
}

.redbutton
{
```

```
background-color: #ffcccc;
}

.button5
{
background-color: #66ff99;
border-bottom:solid;
border-left: #FFEEEE;
border-right:solid;
border-top: #EEEEEE;
color: black;
font-family: Verdana, Arial
}

#off{
 font-family: arial;
 font-size: 11px;
 padding: 2 4 3 2 ;
 border-color: #b8b8b8;
 border-width: 1px;
 background-color: #c0c0c0;
 font-weight: normal;
 -webkit-border-radius: 3 ;
 -moz-border-radius: 3;
 border-radius: 3 ;
}

#on{
 font-family: arial;
 font-size: 11px;
 padding: 2 4 3 2;
 border-color: #b8b8b8;
 border-width: 1px;
 background-color: #d4961b;
 font-weight: normal;
 -webkit-border-radius: 3 ;
 -moz-border-radius: 3 ;
 border-radius: 3 ;
}
</style>
```

E. MENU.PHP

```
<?php
if(isset($documentId)){
 if(isset($profileId)){
```

```
$menuItems=array('import','codefunctions','documents','grou
ps','script','hosts','uploadresults','uploadconfig','scans'
,'configs','reviewscans');
}else{
$menuItems=array('import','codefunctions','documents','host
s','uploadresults','uploadconfig','scans','configs','review
scans');
 }
}else{
$menuItems=array('import','codefunctions','documents','host
s','uploadresults','uploadconfig','scans','configs','review
scans');
}
print "<table class=menu><tr>";
foreach($menuItems as $menuItem){
 $menuItemUrl="$menuItem.php";
 if($menuItem=="script"){
  $menuItem="Generate Scripts";
 }
 print "<td class=menu><a
href=$menuItemUrl>$menuItem</a></td>";
}
print "</tr></table>";
?>
```

F. IMPORT.PHP

```
<?php
include "includes.php";

$mysqli = new mysqli($dbServer,$dbUser,$dbPass,$dbName);
$profileId="";
print "<font face=arial size=2>";
$section="head";
$printSection="group";
$r="<font face=arial color=red size=2><b>";
$bl="<font face=arial color=blue size=3><b>";
$b="<font face=arial color=black size=2>";
$e="</b></font>$b";
$s=" ";
$ID='';

//### DEBUG - Enable Write to DB (0=disable,1=enable)
$documentsInsert=1;
$profilesInsert=1;
$profilesMapInsert=1;
```

```
$groupsInsert=1;

//### DEBUG - Enable Show Vars (0=disable,1=enable)
$showVars=0;

if(isset($_POST['xmlfile'])){
 $xmlfile=$_POST['xmlfile'];
}
if(isset($_POST['deleteAll'])){

$dbTables=array('groups','profilesMap','profiles','document
s');
 foreach($dbTables as $dbTable){
  $sql="truncate $dbTable";
  $mysqli->query($sql);
 }
}
if(!isset($xmlfile)){
 //### Populate Files Array ###
 $files=array();
 if ($handle = opendir('./content')){
  while (false !== ($file = readdir($handle))){
   if (($file!=".")&&($file!="..")){
    $fileExt=substr($file, strrpos($file, '.')+1);
    if($fileExt=="xml"){
     array_push($files, $file);
    }
   }
  }
  closedir($handle);
  sort($files);
  print "<table class=list>";
  print "<tr><td class=listTitle colspan=20>Import
Content</td></tr>";
  $table="documents";
  $docCount=0;
  foreach($files as $file){
   $docCount++;
   $sql2="select COUNT(id) from $table where
xmlfile='$file'";
   if ($result = $mysqli->query($sql2)){
     while ($row = $result->fetch_assoc()){
      $existingRecords=$row['COUNT(id)'];
     }
     mysqli_free_result($result);
   }
```

```
   if($existingRecords<1){
    if($cc==1){
     $tc=$rc1;
     $cc=0;
    }else{
     $tc=$rc2;
     $cc=1;
    }
    print "<tr><td class=list bgcolor=$tc>$docCount</td><td
class=list bgcolor=$tc>$file</td><td class=list
bgcolor=$tc>";
    print "<form action=$phpSelf method=post style=margin-
bottom:0;>";
    print "<input type=hidden name=xmlfile value='$file'>";
    print "<input type=submit value=import></form>";
    print "</td></tr>";
   }else{
    print "<tr><td class=list bgcolor=$tc>$docCount</td><td
class=list bgcolor=$tc>$file</td><td class=list
bgcolor=$tc>";
    print "</td></tr>";
   }
  }
  print "</table>";
 }
//### Parse XML File ##########
}else{
 $xmlfilePath="content/$xmlfile";
 $fp = fopen($xmlfilePath, 'r');
 $xmldata = fread($fp,filesize($xmlfilePath));
 fclose($fp);
 $p = xml_parser_create();
 xml_parse_into_struct($p, $xmldata, $vals, $index);
 xml_parser_free($p);
 foreach($vals as $key=>$val){
  $type='';
  $level='';
  $value='';
  $tag='';
  foreach($val as $key2=>$val2){
   if($showVars==1){
    print "$r key2:$e $key2";
   }
   //### LEVEL 2 ###
   if(!is_array($val2)){
    $$key2=$val2;
```

```
    if($showVars==1){
     print "$r val2:$e $val2<br>";
    }
    if($section=="head"){
      if(($tag=="TITLE")&&($key2=="value")){
      $documentTitle="$val2";
     }
      if(($tag=="DESCRIPTION")&&($key2=="value")){
      $documentDescription="$val2";
     }
      if(($tag=="DC:PUBLISHER")&&($key2=="value")){
      $documentPublisher="$val2";
     }
      if(($tag=="DC:SOURCE")&&($key2=="value")){
      $documentSource="$val2";
     }
     if($ID=="release-info"){
       if(($tag=="PLAIN-TEXT")&&($key2=="value")){
       $documentRelease="$val2";
      }
       if(($tag=="VERSION")&&($key2=="value")){
       $documentReleaseVersion="$val2";
      }
     }
    }elseif($section=="profile"){
      if(($tag=="TITLE")&&($key2=="value")){
      $profileTitle=$val2;
     }
     //### Create Profile Rectord ###
      if(($started==1)&&($tag=="DESCRIPTION")){
      if(isset($documentId)){

$tableVars=array('profileName','profileTitle','documentId')
;
       $table="profiles";
       //### Build SQL Query to add data to Profiles Table
       $sql="insert into $table (";
       $count=1;
       foreach($tableVars as $tableVar){
        if($count>=2){
         $sql.=",";
        }
        $sql.=$tableVar;
        $count++;
       }
       $sql.=") values (";
```

```
        $count=1;
        foreach($tableVars as $tableVar){
         if($count>=2){
          $sql.=",";
         }
         $sql.="\"${$tableVar}\"";
         $count++;
        }
        $sql.=")";
        //### Check for Existing Records (auditName,
statusDate, documentRelease, documentVersion and
benchmarkDate)
        $sql2="select COUNT(id) from $table where
profileName='$profileName' and profileTitle='$profileTitle'
and documentId='$documentId'";
        if ($result = $mysqli->query($sql2)){
         while ($row = $result->fetch_assoc()){
          $existingRecords=$row['COUNT(id)'];
         }
         mysqli_free_result($result);
        }
        //### Execute Query if No Existing Record Exists
        if($existingRecords<1){
         if($profilesInsert==1){
           $mysqli->query($sql);
          }
         }
        $sql="select id from $table where
profileName='$profileName' and profileTitle='$profileTitle'
and documentId='$documentId'";
        if ($result = $mysqli->query($sql)){
         while ($row = $result->fetch_assoc()){
          $profileId=$row['id'];
         }
          mysqli_free_result($result);
        }
       }
        $started=2;
       }
    }elseif($section=="group"){
      if(($key3=="ID")&&($val2=="TITLE")){
      $vulnId=$val3;
     }
      if($key2=="tag"){
      $tag==$val2;
     }
```

```
      if($tag=="VERSION"){
      $version=$val2;
     }
      if($tag=="TITLE"){
      $title=$val2;
     }

if(($tag=="DESCRIPTION")&&($level=="4")&&($val2!="4")){
       //### Parse Description
      $tmpVar="description";
       $descriptionLine=$val2;
       $delimiter="VulnDiscussion";
       $delimiter1="<$delimiter>";
       $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$val2);$$tmpVar=$tmpVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse False Positives
      $tmpVar="falsePositives";
      $delimiter="FalsePositives";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse False Negatives
      $tmpVar="falseNegatives";
      $delimiter="FalseNegatives";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Documentable Status
      $tmpVar="documentable";
      $delimiter="Documentable";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";
```

```
$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Mitigations
      $tmpVar="mitigations";
      $delimiter="Mitigations";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Severity Override Guidance
      $tmpVar="severityOverrideGuidance";
      $delimiter="SeverityOverrideGuidance";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
      //### Parse Potential Impacts
      $tmpVar="potentialImpacts";
      $delimiter="PotentialImpacts";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Third Party Tools
      $tmpVar="thirdPartyTools";
      $delimiter="ThirdPartyTools";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";
```

```
$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Mitigation Controls
      $tmpVar="mitigationControl";
      $delimiter="MitigationControl";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse Responsibility
      $tmpVar="responsibility";
      $delimiter="Responsibility";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
       //### Parse IA Controls
      $tmpVar="iaControls";
      $delimiter="IAControls";
      $delimiter1="<$delimiter>";
      $delimiter2="</$delimiter>";

$tmpVars=explode($delimiter1,$descriptionLine);$$tmpVar=$tm
pVars[1];

$tmpVars=explode($delimiter2,${$tmpVar});$$tmpVar=$tmpVars[
0];
      }
      if($tag=="DC:TITLE"){
      $dcTitle=$val2;
     }
      if($tag=="DC:PUBLISHER"){
      $dcPublisher=$val2;
     }
```

```
   if($tag=="DC:TYPE"){
   $dcType=$val2;
  }
   if($tag=="DC:SUBJECT"){
   $dcSubject=$val2;
  }
   if($tag=="DC:IDENTIFIER"){
   $dcIdentifier=$val2;
  }
   if(($tag=="IDENT")&&($key2=="value")){
   $identCci=$val2;
  }
   if(($tag=="FIXTEXT")&&($key2=="value")){
   $fixText=$val2;
  }
  if(($tag=="CHECK-CONTENT")&&($key2=="value")){
   $checkText=$val2;
  }
 }
}else{
 if($showVars==1){
  print "<br><br>";
 }
 foreach($val2 as $key3=>$val3){
  //### LEVEL 3 ###
  if(!is_array($val3)){
   $$key3=$val3;
    if($showVars==1){
    print "$s$s$s$s$r key3:$e$key3 $r val3:$e$val3<br>";
   }
   if($section=="head"){
    if($key3=="XMLNS:DSIG"){
     $xmlNsDsig=$val3;
    }
    if($key3=="XMLNS:XHTML"){
     $xmlNsXhtml=$val3;
    }
    if($key3=="XMLNS:XSI"){
     $xmlNsXsi=$val3;
    }
    if($key3=="XMLNS:CPE"){
     $xmlNsCpe=$val3;
    }
    if($key3=="XMLNS:DC"){
     $xmlNsDc=$val3;
    }
```

```
        if(($tag=="BENCHMARK")&&($key3=="ID")){
         $xmlId=$val3;
        }
        if($key3=="XML:LANG"){
         $xmlLang=$val3;
        }
        if($key3=="XSI:SCHEMALOCATION"){
         $xsiSchemaLocation=$val3;
        }
        if($key3=="XMLNS"){
         $xmlNs=$val3;
        }
        if($key3=="DATE"){
         $documentDate=$val3;
        }
         if($key3=="HREF"){
         $documentHref="$val3";
        }
       }elseif($section=="profile"){
        if($key3=="ID"){
         $profileName=$val3;
        }
        if($key3=="IDREF"){
         $vulnId=$val3;
        }
        //### Create ProfilesMap Entry ###
         if(($key3=="SELECTED")&&($val3=="true")){
          $tableVars=array('profileId','vulnId');
         $table="profilesMap";
         //### Build SQL Query to add data to ProfilesMap
Table
         $sql="insert into $table (";
         $count=1;
         foreach($tableVars as $tableVar){
          if($count>=2){
           $sql.=",";
          }
          $sql.=$tableVar;
          $count++;
         }
         $sql.=") values (";
         $count=1;
         foreach($tableVars as $tableVar){
          if($count>=2){
           $sql.=",";
          }
```

```
         $sql.="\"${$tableVar}\"";
         $count++;
        }
        $sql.=")";
        //### Check for Existing Records (auditName,
statusDate, documentRelease, documentVersion and
benchmarkDate)
        $sql2="select COUNT(id) from $table where
profileId='$profileId' and vulnId='$vulnId'";
        if ($result = $mysqli->query($sql2)){
         while ($row = $result->fetch_assoc()){
          $existingRecords=$row['COUNT(id)'];
         }
        }
         mysqli_free_result($result);
        //### Execute Query if No Existing Record Exists
        if($existingRecords<1){
          if($profilesMapInsert==1){
            $mysqli->query($sql);
           }
         }
        $sql="select id from $table where
profileId='$profileId' and vulnId='$vulnId'";
        if ($result = $mysqli->query($sql)){
         while ($row = $result->fetch_assoc()){
          $profilesMapId=$row['id'];
         }
          mysqli_free_result($result);
        }
        $started=2;
        }
       }elseif($section=="group"){
        if(($tag=="RULE")&&($key3=="ID")){
        $ruleId=$val3;
       }
        if(($tag=="RULE")&&($key3=="SEVERITY")){
        $severity=$val3;
       }
        if(($tag=="RULE")&&($key3=="WEIGHT")){
        $weight=$val3;
       }
        if(($tag=="IDENT")&&($key3=="SYSTEM")){
        $identSystemUrl=$val3;
       }
        if(($tag=="FIXTEXT")&&($key3=="FIXREF")){
        $fixRefId=$val3;
```

```
        }
         if(($tag=="FIX")&&($key3=="ID")){
         $fixId=$val3;
        }
         if(($tag=="CHECK")&&($key3=="SYSTEM")){
         $chkId=$val3;
        }
         if(($tag=="CHECK-CONTENT-REF")&&($key3=="NAME")){
         $checkContentRef=$val3;
         print "-$val3-<br>";
        }
         if(($tag=="CHECK-CONTENT-REF")&&($key3=="HREF")){
         $checkContentHref=$val3;
        }
        }
       }
     }
    }

if(($section!="head")&&($tag=="BENCHMARK")&&($type=="open")
){
     $section="head";
    }
    //### Create Documents Entry ###
    if(($section=="head")&&($tag=="PROFILE")){
     $section="profile";
     $started=0;

$tableVars=array('xmlNsDsig','xmlNsXhtml','xmlNsXsi','xmlNs
Cpe','xmlNsDc','xmlId','xmlLang','xsiSchemaLocation','xmlNs
','documentDate','documentTitle','documentDescription','doc
umentPublisher','documentSource','documentHref','documentRe
lease','documentReleaseVersion','xmlfile');
     $table="documents";
     //### Build SQL Query to add data to documents Table
     $sql="insert into $table (";
     $count=1;
     foreach($tableVars as $tableVar){
      if($count>=2){$sql.=",";}
      $sql.=$tableVar;
      $count++;
     }
     $sql.=") values (";
     $count=1;
     foreach($tableVars as $tableVar){
      if($count>=2){
```

```
      $sql.=",";
     }
     $sql.="\"${$tableVar}\"";
     $count++;
    }
    $sql.=")";
    //### Check for Existing Records (auditName,
statusDate, documentRelease, documentVersion and
benchmarkDate)
    $sql2="select COUNT(id) from documents where
xmlId='$xmlId' and documentDate='$documentDate' and
documentTitle='$documentTitle' and
documentDescription='$documentDescription' and
documentRelease='$documentRelease' and
documentReleaseVersion='$documentReleaseVersion' and
xmlfile='$xmlfile'";
    if ($result = $mysqli->query($sql2)){
     while ($row = $result->fetch_assoc()){
      $existingRecords=$row['COUNT(id)'];
     }
    }
    mysqli_free_result($result);
    //### Execute Query if No Existing Record Exists
    if($existingRecords<1){
      if($documentsInsert==1){
      $mysqli->query($sql);
      }
    }
    $sql="select id from documents where xmlId='$xmlId' and
documentDate='$documentDate' and
documentTitle='$documentTitle' and
documentDescription='$documentDescription' and
documentRelease='$documentRelease' and
documentReleaseVersion='$documentReleaseVersion' and
xmlfile='$xmlfile'";
    if ($result = $mysqli->query($sql)){
     while ($row = $result->fetch_assoc()){
      $documentId=$row['id'];
     }
    }
    mysqli_free_result($result);
    print "documentId: $documentId - $xmlfile<br>";
   }

if(($section=="profile")&&($tag=="PROFILE")&&($type=="open"
)&&($started==0)){
```

```
    $started=1;
   }

if(($section=="profile")&&($tag=="PROFILE")&&($type=="close
")&&($started>=1)){
    $started=0;
   }
   if(($section=="profile")&&($tag=="GROUP")){
    $section="group";
    //print "PROFILES FINISHED<br><br>";
    $started=0;
   }

if(($section=="group")&&($tag=="GROUP")&&($type=="open")&&(
$started==0)){
    $started=1;
   }
   //### Create Group Record ###

if(($section=="group")&&($tag=="GROUP")&&($type=="close")&&
($started==1)){
    $started=0;
    $description=$mysqli->real_escape_string($description);
    $title=$mysqli->real_escape_string($title);
    $fixText=$mysqli->real_escape_string($fixText);
    $checkText=$mysqli->real_escape_string($checkText);

$tableVars=array('vulnId','ruleId','severity','weight','ver
sion','title','description','falsePositives','falseNegative
s','documentable','mitigations','severityOverrideGuidance',
'potentialImpacts','thirdPartyTools','mitigationControl','r
esponsibility','iaControls','dcTitle','dcPublisher','dcType
','dcSubject','dcIdentifier','identSystemUrl','identCci','f
ixRefId','fixText','fixId','chkId','checkContentRef','check
ContentHref','checkText');
    $table="groups";
    //### Build SQL Query to add data to Groups Table
    $sql="insert into $table (";
    $count=1;
    foreach($tableVars as $tableVar){
     if($count>=2){
      $sql.=",";
     }
     $sql.=$tableVar;
     $count++;
    }
```

```
    $sql.=") values (";
    $count=1;
    foreach($tableVars as $tableVar){
     if($count>=2){
      $sql.=",";
     }
     $sql.="\"${$tableVar}\"";
     $count++;
    }
    $sql.=")";
    //### Check for Existing Records (auditName,
statusDate, documentRelease, documentVersion and
benchmarkDate)
    $sql2="select COUNT(id) from $table where
vulnId='$vulnId'";
    if ($result = $mysqli->query($sql2)){
     while ($row = $result->fetch_assoc()){
      $existingRecords=$row['COUNT(id)'];
     }
    }
    mysqli_free_result($result);
    //### Execute Query if No Existing Record Exists
    if($existingRecords<1){
     if($groupsInsert==1){
      $mysqli->query($sql);
     }
    }
    $sql="select id from $table where vulnId='$vulnId'";
    if ($result = $mysqli->query($sql)){
     while ($row = $result->fetch_assoc()){
      $groupId=$row['id'];
     }
    mysqli_free_result($result);
    }
   }
  }
 }
mysqli_close($mysqli);
print "
<form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=submit value='Check for more'>
</form>";
}
print "INFO: This page is used for importing Manual and
Benchmark XCCDF XML Content. First place the file in the
content directory.";
```

```
?>
```

G. CODEFUNCTIONS.PHP

```
<?php
include "includes.php";
$dbTable="codeFunctions";
$vars=getFields($dbTable);

//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){
  $$var=$_POST["$var"];
 }
 if(isset($_GET["$var"])){
  $$var=$_GET["$var"];
 }
 if(!isset(${$var})){
  $$var='';
 }
}

//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
}

//*** Add ***
if($mode=="add"){
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
   if($var=="code"){
    $$var=addslashes(${$var});
   }
```

```
  if($count>=2){
   $sql.=",";
  }
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}

//*** Update Database ***
if($mode=="update"){
 $sql="update $dbTable set ";
 $count=1;
 foreach($vars as $var){
  if($var=="code"){
   $$var=addslashes(${$var});
  }
  if($count>=2){
   $sql.=",";
  }
  $sql.="$var=\"${$var}\"";
  $count++;
 }
 $sql.=" where id=$id";
 $mysqli->query($sql);
 $mode="none";
}

//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){
  $$var='';
 }
}

//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
```

```
}

//*** Form Header ***
print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2 class=formtitle>Code
Functions</td></tr>";

//*** Form ***
print "<tr><td class=formlabel>Name:</td><td
class=formfield><input type=text size=80 name=name
value='$name'></td></tr>";
print "<tr><td class=formlabel>Description:</td><td
class=formfield><textarea rows=5 cols=80
name=description>$description</textarea></td></tr>";
print "<tr><td class=formlabel>Code:<br><br><br>
Not A Finding - 0<br>
Open - 1<br>
Manual Check - 2<br>
Exception - 3<br>
Unknown - 4<br>


</td><td class=formfield><textarea rows=20 cols=80
name=code style='color: white; background-color: black'>";
print $code;
print "</textarea></td></tr>";
print "<tr><td class=formlabel>Variables:</td><td
class=formfield><input type=text size=80 name=variables
value='$variables'></td></tr>";
print "<tr><td class=formLabel>Code Type:</td><td
class=formField>";
$sId="codeTypeId";
$qTable="codeTypes";
$qId="id";
$qId2="qid";
$qDisplay="type";
print "<select name=$sId>";
$sql2="select $qId,$qDisplay from $qTable";
$result2 = $mysqli->query($sql2);
while ($row2 = $result2->fetch_assoc()){
 $$qId2=$row2[$qId];
 $$qDisplay=$row2[$qDisplay];
 if(${$qId2}==${$sId}){
```

```
  print "<option selected
value='${$qId2}'>${$qDisplay}</option>";
 }else{
  print "<option value='${$qId2}'>${$qDisplay}</option>";
 }
}
print "</select>";
print "</td></tr>";
print "<tr><td class=formlabel>Tested:</td><td
class=formfield><input type=text size=20 name=tested
value='$tested'></td></tr>";
if ($execute=="on"){
 $executeChecked="checked";
}else{
 $executeChecked="";
}
print "<tr><td class=formlabel>Execute:</td><td
class=formfield><input type=checkbox name=execute
$executeChecked></td></tr>";
print "<tr><td class=formlabel>Creator:</td><td
class=formfield><input type=text size=20 name=creatorId
value='$creatorId'></td></tr>";

//*** Form Footer ***
if($mode=="none"){
 $mode="add";
}
if($mode=="edit"){
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";

//*** BROWSE ***
print "<table class=list>";
```

```
print "<tr>";
$browseVars=array('id','name','type');
foreach($browseVars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "</tr>";

$sql="select cf.id,cf.name,cft.type from codeFunctions cf
join codeTypes cft on (cft.id=cf.codeTypeId)";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  print "<tr>";
  $col=1;
  foreach($browseVars as $var){
   $$var=$row["$var"];
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc>$id <a
href=$phpSelf?mode=edit&id=$id>${$var}</a></td>";
    }else{
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
print "INFO: This page is for creating code functions that
will be used in the edit groups page.";
?>
```

H. DOCUMENTS.PHP

```
<?php
include "includes.php";
$dbTable="documents";
$vars=array('id','documentTitle','xmlfile');
//### Get Variables ###
foreach($vars as $var){
 if(isset($_POST["$var"])){$$var=$_POST["$var"];}
 if(isset($_GET["$var"])){$$var=$_GET["$var"];}
}
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){$$var='';}
}
//### BROWSE ###
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "<td class=listheader></td>";
print "</tr>";
$sql="select * from $dbTable";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;$cc=0;
  }else{
   $tc=$rc2;$cc=1;
  }
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
   if($var!="id"){
     print "<td class=list bgcolor=$tc>${$var}</td>";
   }
  }
  print "<td class=list bgcolor=$tc>
<form action=profiles.php method=post style=margin-
bottom:0;>
<input type=hidden name=setDocumentId value='$id'>
<input type=submit value=select>
```

```
</form></td>";
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
?>
```

I. PROFILES.PHP

```
<?php
include "includes.php";
$dbTable="profiles";
$vars=getFields($dbTable);
//### BROWSE ###
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if(($var!="id")&&($var!="documentId")){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "<td class=listheader></td>";
print "</tr>";
$sql="select * from $dbTable where
documentId='$documentId'";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;$cc=0;
  }else{
   $tc=$rc2;$cc=1;
  }
  print "<tr>";
  foreach($vars as $var){
   $$var=$row["$var"];
   if(($var!="id")&&($var!="documentId")){
    print "<td class=list bgcolor=$tc>${$var}</td>";
   }
  }
  print "<td class=list bgcolor=$tc>
<form action=groups.php method=post style=margin-bottom:0;>
<input type=hidden name=setProfileId value='$id'>
<input type=submit value=select>
```

```
</form></td>";
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
?>
```

J. GROUPS.PHP

```
<?php
include "includes.php";
$dbTable="groups";
$tested='';
$vars=getFields($dbTable);
$browseVars=array('id','vulnId','version','severity','title
');
//### BROWSE ###
$sql="select distinct(pm.vulnId) vulnId,g.id
gGroupId,c.groupId cGroupId,
g.version,g.title,g.severity,g.id, c.bug bug, c.tested
tested, c.id codeId from profilesMap pm join profiles p on
(p.id = pm.profileId) join groups g on (pm.vulnId =
g.vulnId) left join code c on (c.groupId=g.id) where
p.documentId='$documentId' and p.id='$profileId' and c.id
is null";
$result = $mysqli->query($sql);
$remainingRecords=mysqli_num_rows($result);
print "<table class=list>";
print "<tr>";
print "<td class=listHeader>$remainingRecords</td>";
print "<td class=listHeader>Vuln ID</td>";
print "<td class=listHeader>Version</td>";
print "<td class=listHeader>CAT</td>";
print "<td class=listHeader>Title</td>";
print "</tr>";
$count=0;
$sql="select distinct(pm.vulnId) vulnId,g.id
gGroupId,c.groupId cGroupId,
g.version,g.title,g.severity,g.id, c.bug bug, c.tested
tested, c.id codeId from profilesMap pm join profiles p on
(p.id = pm.profileId) join groups g on (pm.vulnId =
g.vulnId) left join code c on (c.groupId=g.id) where
p.documentId='$documentId' and p.id='$profileId'";
if ($result = $mysqli->query($sql)){
```

```
 while ($row = $result->fetch_assoc()){
  $count++;
  if($cc==1){
   $tc=$rc1;$cc=0;
  }else{
   $tc=$rc2;$cc=1;
  }
  print "<tr>";
  $id=$row['id'];
  $cGroupId=$row['cGroupId'];
  $bug=$row['bug'];
  $tested=$row['tested'];
  if($cGroupId){
   $button="yellowbutton";
  }else{
   $button="greybutton";
  }
  if($tested=="on"){
   $button="greenbutton";
  }
  if($bug=="on"){
   $button="redbutton";
  }
  print "<td class=list bgcolor=$tc>
<form action=editgroup.php method=post style=margin-
bottom:0;>
<input type=hidden name=groupId value='$id'>
<input type=submit value=select class=$button>
</form></td>";
  $col=1;
  foreach($browseVars as $var){
   $$var=$row["$var"];
   if($var=="severity"){
    if($severity=="low"){
     $severity="III";$tc2="green";
    }
    if($severity=="medium"){
     $severity="II";$tc2="yellow";
    }
    if($severity=="high"){
     $severity="I";$tc2="red";
    }
   }
   if(($var!="id")&&($var!="tested")){
    if($var=="severity"){
     $myColor=$tc2;
```

```
     print "<td class=list align=center
bgcolor=$myColor>${$var}</td>";
    }else{
     $myColor=$tc;
     print "<td class=list bgcolor=$myColor>${$var}</td>";
    }
   }
  }
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
print "Records: $count<br>";
?>
```

K. EDITGROUP.PHP

```
<?php
include "includes.php";
$status='';
$code='';
$codeId='';
$codeFunctionId='';
$bug='';
$tested='';
$notes='';

//*** Update Database ***
if($mode=="update"){
 $code=addslashes($_POST['code']);
 $codeId=$_POST['codeId'];
 $notes=$_POST['notes'];

if(isset($_POST['tested'])){$tested=$_POST['tested'];}else{
$tested='';}

if(isset($_POST['bug'])){$bug=$_POST['bug'];}else{$bug='';}
 $codeFunctionId=$_POST['codeFunctionId'];
 $sql="update code set bug='$bug',tested='$tested',
code='$code',fnId='$codeFunctionId',notes='$notes' where
id='$codeId'";
 $mysqli->query($sql);
 $mode="edit";
}
```

```
//*** Delete Record ***
if($mode=="delete"){
 $codeId=$_POST['codeId'];
 $sql="delete from code where id='$codeId'";
 $result = $mysqli->query($sql);
 $mode="none";
}
if(!isset($_POST['groupId'])){
 print "Please access this page from the groups page.<br>";
 exit;
}else{
 $groupId=$_POST['groupId'];
 $dbTable="groups";
 $vars=getFields($dbTable);
 $sql="select * from $dbTable where id=$groupId";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
 $sql="select documentTitle from documents where
id='$documentId'";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  $documentTitle=$row['documentTitle'];
 }
 $result->close();
 $today=date('m/d/Y');
 $codeHeader="#!/bin/bash
# DATE: $today
# CHECK: $version
# VULN: $vulnId
# TITLE: $title
# \$status: 0=not a finding. 1=open finding. 2=manual
check. 3=unable to check. 4=unknown.
";

 //*** Insert Function ***

if(($mode=="insertFunction")||($mode=="insertFunctionWizard
")){
  if($mode=="insertFunction"){
   $codeFunctionId=$_POST['codeFunctionId'];
```

```
  }
  if($codeFunctionId){

   //*** Get function ***
   $groupId=$_POST['groupId'];
   $sql="select code,variables,execute from codeFunctions
where id=$codeFunctionId";
   $result=$mysqli->query($sql);
   while ($row = $result->fetch_assoc()){
    $code=$row['code'];
    $execute=$row['execute'];
    $variables=$row['variables'];
   }
   $result->close();
  }

  //*** End of checkText matching ***
  $sql="select id codeId from code where groupId=$groupId";
  $result=$mysqli->query($sql);
  while ($row = $result->fetch_assoc()){
   $codeId=$row['codeId'];
  }
  if($codeId>=1){
   $mode="edit";
  }else{
   $mode="none";
  }
  $result->close();
 }else{
  $codeFunctionId='';
  $sql="select * from code where groupId='$groupId'";
  $result=$mysqli->query($sql);
  while ($row = $result->fetch_assoc()){
   $codeId=$row['id'];
   $tested=$row['tested'];
   $bug=$row['bug'];
   $notes=$row['notes'];
   $code=$row['code'];
   $codeFunctionId=$row['fnId'];
  }
  $result->close();
  if($code){$mode="edit";}
 }

 //*** Add ***
 if($mode=="add"){
```

```
  if(isset($_POST['tested'])){
   $tested=$_POST['tested'];
  }
  if(isset($_POST['bug'])){
   $bug=$_POST['bug'];
  }
  if(isset($_POST['notes'])){
   $notes=$_POST['notes'];
  }
  $code=addslashes($_POST['code']);
  $sql="insert into code
(groupId,code,fnId,tested,bug,notes) values
('$groupId','$code','$codeFunctionId','$tested','$bug','$no
tes')";
  $mysqli->query($sql);
  $mode="edit";
  $sql="select * from code where groupId='$groupId'";
  $result=$mysqli->query($sql);
  while ($row = $result->fetch_assoc()){
   $codeId=$row['id'];
   $codeFunctionId=$row['fnId'];
  }
  $result->close();
 }
if($code==""){$code=$codeHeader . "

echo \$status\$notes";
}
$prevGroupId=$groupId-1;
$nextGroupId=$groupId+1;
print "
<table cellpadding=0 cellspacing=0><tr><td>
<form action=editgroup.php method=post style=margin-
bottom:0;>
<input type=hidden name=groupId value='$prevGroupId'>
<input type=submit value='<'>
</form>
</td><td>
<form action=editgroup.php method=post style=margin-
bottom:0;>
<input type=hidden name=groupId value='$nextGroupId'>
<input type=submit value='>'>
</form>
</td></tr>
</table>
";
```

```
if($severity=="low"){
 $severityCat="CAT III";
}
elseif($severity=="medium"){
 $severityCat="CAT II";
}
elseif($severity=="high"){
 $severityCat="CAT I";
}
else{
 $severityCat=$severity;
}

 //*** Display groups info for this groupId ***
 $p1="<pre style='white-space:pre-wrap;'><font
face=arial>";
 $p2="</font></pre>";
 print "<table width=900><tr><td>"; // table surrounding
the 2 sections
 print "<table width=100%>";
 $httpReferer=$_SERVER['HTTP_REFERER'];
 $lastElement=basename($_SERVER['SCRIPT_NAME']);

$groupsPage=preg_replace("!".$lastElement."!",'groups.php',
$httpReferer);
 print "<tr><td class=formLabel><p id=tooltip2><a
href=$groupsPage>Title:<span style='text-align:left;white-
space:normal;width:600;'><b>DESCRIPTION:</b><BR>$descriptio
n</span></a></p></td><td width=100%
class=formfieldsmall>$version $vulnId $title</td></tr>";
 print "<tr><td class=formLabel><p id=tooltip2><a
href=$groupsPage>Check:<span style='text-align:left;white-
space:pre-
wrap;width:600;'><b>FIX:</b><BR>$fixText</span></a></p></td
><td class=formfield>$p1$checkText$p2</td></tr>";
 print "<tr><td class=formLabel>Tresys:</td><td
class=formfieldsmall>";
 print "<a target='_none'
href=http://oss.tresys.com/projects/clip/browser/packages/a
queduct/aqueduct/compliance/Bash/STIG/rhel-
5/prod/$version.sh>";
 print "Tresys Link</a></td></tr>";
 print "</table>";
 print "</td></tr><tr><td>"; //separate the 2 sections
```

```
 //*** Check/Audit Code ***
 print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form width=100%>";
 print "<tr><td class=formlabel>Functions:</td><td>";
 print "<table><tr><td>";
 $sId="codeFunctionId";
 $qTable="codeFunctions";
 $qId="id";
 $qId2="qid";
 $qDisplay="name";
 print "<select name=$sId>";
 $sql2="select $qId,$qDisplay from $qTable";
 $result2 = $mysqli->query($sql2);
 print "<option value=''>--None--</option>";
 while ($row2 = $result2->fetch_assoc()){
  $$qId2=$row2[$qId];
  $$qDisplay=$row2[$qDisplay];
  if(${$qId2}==${$sId}){
   print "<option selected
value='${$qId2}'>${$qDisplay}</option>";
  }else{
  print "<option value='${$qId2}'>${$qDisplay}</option>";
  }
 }
 print "
 </select>
 <input type=hidden name='codeId' value='$codeId'>
 <input type=hidden name='groupId' value='$groupId'>
 <input type=hidden name='mode' value='insertFunction'>
<input type=submit value='Insert'>
 </form>";


 print "</td></tr></table></td></tr>";

 if($tested=="on"){
  $testedChecked="checked";
 }else{
  $testedChecked='';
 }
 if($bug=="on"){
  $bugChecked="checked";
 }else{
  $bugChecked='';
 }
```

```
 print "<form action=$phpSelf method=post style=margin-
bottom:0;>";
 print "<tr><td class=formLabel>Status:</td><td
class=formfield>";
 print "Tested: <input name=tested type=checkbox
$testedChecked>    ";
 print "Bug: <input name=bug type=checkbox $bugChecked>";
 print "</td></tr>";

 print "<tr><td class=formlabel>Notes:</td><td
class=formfield>";
 print "<textarea rows=2 cols=100
name=notes>$notes</textarea></td></tr>";
 print "<tr><td class=formlabel>Code:</td><td
class=formfield>";
 print "<textarea rows=30 cols=114 name=code style='color:
white; background-color:
black'>$code</textarea></td></tr>";
 print "<input type=hidden name=groupId value=$groupId>";

 //*** Form Footer ***
 if($mode=="none"){
  $mode="add";
 }
 if($mode=="edit"){
  $mode="update";
 }
 print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
 print "<td>
  <input type=hidden name=codeId value='$codeId'>
  <input type=hidden name=codeFunctionId
value='$codeFunctionId'>

<input type=hidden name=mode value=$mode><input type=submit
value=$mode></form></td>";
 if($mode=="update"){
  print "
  <form action=$phpSelf method=post style=margin-bottom:0;>
  <input type=hidden name=groupId value='$groupId'>
  <input type=hidden name=codeId value='$codeId'>
  <input type=hidden name=mode value=delete>
  <td><input type=submit value=delete></td></form>";
 }
 print "</tr></table></td></tr></table>";
}
```

```
print "</td></tr></table>"; //close off the 2 sections
print "<table class=form>";
print "<tr><td>
<form action=$phpSelf method=post style=margin-bottom:0;>
<input type=hidden name=scannow value=yes>
<input type=hidden name=groupId value='$groupId'>
<input type=submit value=scan></form>

</td></tr>";
if(isset($_POST['scannow'])){
 $scriptDir="scanbox";
 $today = date("d-M-Y Hi");
 $wrapperFile="$scriptDir/runall.sh";
 $wrapperHandle = fopen($wrapperFile, 'w') or die("can't
open file");
 $wrapperHeader="#!/bin/bash
# SuperSCAP Wrapper
hostname=`hostname`
osType=`uname -s`
report(){
 if [ \"\$osType\" = \"SunOs\" ];then
  startTime=`/usr/bin/truss /usr/bin/date 2>&1 | nawk -F=
'/^time\(\)/ {gsub(/ /,\"\",$2);print $2}'`
 else
  startTime=`date +%s`
 fi
 line=`./\$version.sh`
 exitCode=$?
 if [ \"\$osType\" = \"SunOs\" ];then
  endTime=`/usr/bin/truss /usr/bin/date 2>&1 | nawk -F=
'/^time\(\)/ {gsub(/ /,\"\",$2);print $2}'`
 else
  endTime=`date +%s`
 fi
 totalTime=`expr \$endTime - \$startTime`
 if [ \$totalTime -gt 10 ];then
  echo \$version >> slow_scripts
 fi
 if [ \$exitCode -eq 0 ];then
  status=`echo \$line |cut -c1`
  notes=`echo \$line |cut -c2-`
  echo \$version >> ok_scripts
 else
  status=unknown
  notes='script could not run properly'
  echo \$version.sh had an issue
```

```
  echo \$version.sh >> problem_scripts
 fi
 echo;echo \"VULN ID: \$vulnId  VERSION: \$version  STATUS:
\$status\"
 echo \"TITLE: \$title\"
 echo -e \"NOTES: \$notes\"
 echo
\"\$vulnId;\$version;\$status;\$title;\$vc;\$vo;\$notes\"
>> \$hostname.log
}
";
 fwrite($wrapperHandle, $wrapperHeader);
 $thisFile="$scriptDir/$version.sh";
 $thisHandle = fopen($thisFile, 'w') or die("can't open
file");
 fwrite($thisHandle, $code);
 fclose($thisHandle);
 `dos2unix $thisFile`;
 chmod($thisFile,0777);

$thisScript="vulnId='$vulnId';version=$version;title='$titl
e';report";
 fwrite($wrapperHandle, $thisScript);
 fclose($wrapperHandle);
 `dos2unix $wrapperFile`;
 chmod($wrapperFile,0777);
 $sessionConfigId=$_SESSION['configId'];
 $sql="select file from configs where id=$sessionConfigId";
 $result = $mysqli->query($sql);
 $row = $result->fetch_assoc();
 $configFile=$row['file'];
 copy("$configFile","scanbox/device.cfg");
 chdir("scanbox");
 $output=`./runall.sh`;
 chdir("../");
 print "<tr><td><pre>$output</pre></td></tr>";
}
print "</table>";
?>
```

L. SCRIPT.PHP

```
<?php
include "includes.php";
$hostId='';
$platform='';
```

```
$b="<br>";
//*** Clear Script DIR ***
$scriptDir="superscap";
`rm $scriptDir/*.sh`;
`rm $scriptDir/*.log`;
`rm $scriptDir/*.csv`;
`rm $scriptDir/*.xml`;
`rm $scriptDir/*.html`;
`rm $scriptDir/*.txt`;
`rm $scriptDir/*.gz`;

//*** SQL Query for Custom Check Scripts ***
$sql="select distinct(pm.vulnId) vulnId,g.ruleId ruleId,
g.description gDescription, g.severity gSeverity,
g.checkText gCheckText, g.fixText gFixText, g.id
gGroupId,c.code code,c.groupId cGroupId, g.version
version,g.title title,g.severity severity,g.id, c.id codeId
from profilesMap pm join profiles p on (p.id =
pm.profileId) join groups g on (pm.vulnId = g.vulnId) left
join code c on (c.groupId=g.id) where
p.documentId='$documentId' and p.id='$profileId'";
if ($result = $mysqli->query($sql)){
 $today = date("d-M-Y Hi");
 $wrapperFile="$scriptDir/runall.sh";
 $wrapperHandle = fopen($wrapperFile, 'w') or die("can't
open file");
 //*** Create Wrapper ***
 $wrapperHeader="#!/bin/bash
# SuperSCAP Wrapper
hostname=`hostname`
osType=`uname -s`
report(){
 if [ \"\$osType\" = \"SunOs\" ];then
  startTime=`/usr/bin/truss /usr/bin/date 2>&1 | nawk -F=
'/^time\(\)/ {gsub(/ /,\"\",$2);print $2}'`
 else
  startTime=`date +%s`
 fi
 line=`./\$version.sh`
 exitCode=$?
 if [ \"\$osType\" = \"SunOs\" ];then
  endTime=`/usr/bin/truss /usr/bin/date 2>&1 | nawk -F=
'/^time\(\)/ {gsub(/ /,\"\",$2);print $2}'`
 else
  endTime=`date +%s`
 fi
```

```
 totalTime=`expr \$endTime - \$startTime`
 if [ \$totalTime -gt 10 ];then
  echo \$version >> slow_scripts
 fi
 if [ \$exitCode -eq 0 ];then
  status=`echo \$line |cut -c1`
  notes=`echo \$line |cut -c2-`
  echo \$version >> ok_scripts
 else
  status=unknown
  notes='script could not run properly'
  echo \$version.sh had an issue
  echo \$version.sh >> problem_scripts
 fi
 echo;echo \"RULE ID: \$ruleId  VULN ID: \$vulnId  VERSION:
\$version  STATUS: \$status\"
 echo \"TITLE: \$title\"
 echo -e \"NOTES: \$notes\"
 echo
\"\$ruleId;\$vulnId;\$version;\$status;\$title;\$vc;\$vo;\$
notes\" >> \$hostname.log
}
";
 //*** Write Script File ***
 fwrite($wrapperHandle, $wrapperHeader);
 while ($row = $result->fetch_assoc()){
  $ruleId=$row['ruleId'];
  $vulnId=$row['vulnId'];
  $version=$row['version'];
  $code=$row['code'];
  $title=str_replace('\'','\\\'',$row['title']);
  $severity=$row['severity'];
  $gDescription=$row['gDescription'];
  $gFixText=$row['gFixText'];
  $gCheckText=$row['gCheckText'];
  $cGroupId=$row['cGroupId'];
  if($severity=="high"){$severity="CAT I";}
  if($severity=="medium"){$severity="CAT II";}
  if($severity=="low"){$severity="CAT III";}
  if($cGroupId){
   $thisFile="$scriptDir/$version.sh";
   $thisHandle = fopen($thisFile, 'w') or die("can't open
file");
   fwrite($thisHandle, $code);
   fclose($thisHandle);
   `dos2unix $thisFile`;
```

```
   chmod($thisFile,0777);

$thisScript="vulnId='$vulnId';ruleId=$ruleId;version=$versi
on;title='$title';report
";
  print "adding $version<br>";
  fwrite($wrapperHandle, $thisScript);
  }
 }
fclose($wrapperHandle);
`dos2unix $wrapperFile`;
chmod($wrapperFile,0777);
}

//*** Compress Scripts ***
system("tar --exclude=SuperSCAPScripts.tar.gz -czf
superscap/SuperSCAPScripts.tar.gz superscap 2> /dev/null");

print "Scripts have been generated.<br>";
print "Click <a
href=superscap/SuperSCAPScripts.tar.gz><b>HERE</b></a> to
download.";

print "<table class=form>";
print "
<form action=$phpSelf method=post style=margin-bottom:0;>

<tr><td class=formtag>Host:</td><td class=formfield>
<select name=hostId>
<option value=''>--SELECT--</option>
";
$sql="select id,name from hosts";
$result = $mysqli->query($sql);
while ($row = $result->fetch_assoc()){
 $hostId=$row['id'];
 $hostName=$row['name'];
 print "<option value=$hostId>$hostName</option>";
}
print "
</select></td></tr>
<tr><td class=formtag>Platform:</td><td
class=formfield><input type=text size=40
name=platform></td></tr>

<input type=hidden name=scannow value=yes>
```

```
<tr><td align=center colspan=2><input type=submit
value=scan></form></td></tr>

</td></tr>";

//*** Execute Scripts ***
if(isset($_POST['scannow'])){
 $scriptDir="scanbox/superscap";
 $sessionConfigId=$_SESSION['configId'];
 $sql="select file from configs where id=$sessionConfigId";
 $result = $mysqli->query($sql);
 $row = $result->fetch_assoc();
 $configFile=$row['file'];
 chdir("scanbox");
 $myDir=getcwd();
 $outHost=gethostname();
 $outFile=$myDir . "/superscap/" . $outHost . ".log";
 unlink($outFile);
 `tar -zxvf ../superscap/SuperSCAPScripts.tar.gz`;
 chdir("../");
 copy("$configFile","$scriptDir/device.cfg");
 chdir($scriptDir);
 $output=`./runall.sh`;
 chdir("../../");
 print "<tr><td><pre>$output</pre></td></tr>";
}
print "</table>";

//*** Parse Scan Output ***
if(isset($_POST['scannow'])){
 $outHost=gethostname();
 $outFile="scanbox/superscap/" . $outHost . ".log";
 $handle = fopen($outFile, "r");
 if($handle){
  //*** Create Scan Entry ***
  $myHostId=$_POST['hostId'];
  $myPlatform=$_POST['platform'];
  $sql="insert into scans (hostId,timestamp,file,platform)
values ('$myHostId','$now','$outFile','$myPlatform')";
  $mysqli->query($sql);
  //*** Get ID of Scan Entry ***
  $result=$mysqli->query("select id from scans order by id
desc limit 1");
  $row = $result->fetch_assoc();
  $scanId=$row['id'];
  while (($line = fgets($handle)) !== false){
```

```
   $tmpVar=$line;
   $tmpVars=explode(";",$tmpVar);
   $myRuleId=$tmpVars[0];
   $myStatus=$tmpVars[3];
   $myNotes=$tmpVars[7];
   $identCci='';

if($myStatus=="0"){$myResult="pass";}else{$myResult="fail";
}
   //*** Insert Results into DB ***
   $sql="insert into results
(ruleId,result,identCci,timestamp,scanId) values
('$myRuleId','$myResult','$identCci','$now','$scanId')";
   $mysqli->query($sql);
  }
 }else{
 print "Could not open $outFile<br>";
 }
}
?>
```

M. HOSTS.PHP

```
<?php
include "includes.php";
$dbTable="hosts";
$vars=getFields($dbTable);
//### Get Variables ###
foreach($vars as $var){
 if(isset($_POST["$var"])){$$var=$_POST["$var"];}
 if(isset($_GET["$var"])){$$var=$_GET["$var"];}
}
//### Delete Record ###
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
}
//### Add ###
if($mode=="add"){
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.=$var;
  $count++;
```

```
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}
//### Update Database ###
if($mode=="update"){
 $sql="update $dbTable set ";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.="$var=\"${$var}\"";
  $count++;
 }
 $sql.=" where id=$id";
 $mysqli->query($sql);
 $mode="none";
}
//### Define Variables ###
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){$$var='';}
}
//### Query DB for Edit ###
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
}
//### Form Header ###
print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";
//### Form ###
```

```
foreach($vars as $var){
 $uc_var=ucfirst($var);
 if($var!="id"){
  print "<tr><td class=formtag>$uc_var:</td><td
class=formfield><input type=text size=20 name=$var
value='${$var}'></td></tr>";
 }
}
//### Form Footer ###
if($mode=="none"){$mode="add";}
if($mode=="edit"){
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";
//### BROWSE ###
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "</tr>";
$sql="select * from $dbTable";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
```

```
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc><a
href=$phpSelf?mode=edit&id=$id>${$var}</a></td>";
    }else{
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
?>
```

N. UPLOADRESULTS.PHP

```
<?php
include "includes.php";
$dbTable="scans";
$vars=getFields($dbTable);
$now=time();
$timestamp=time();

//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){
  $$var=$_POST["$var"];
 }
 if(isset($_GET["$var"])){
  $$var=$_GET["$var"];
 }
}

//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
```

```
}

//*** Add ***
if($mode=="add"){

 //*** Upload File ***
 $allowedExts = array("xml");
 $temp = explode(".", $_FILES["file"]["name"]);
 $extension = end($temp);
 if (($_FILES["file"]["size"] < 2000000)
 && in_array($extension, $allowedExts)){
  if (!$_FILES["file"]["error"] > 0){
   move_uploaded_file($_FILES["file"]["tmp_name"],
   "uploads/" . $now . "-" . $_FILES["file"]["name"]);
   $file="uploads/" . $now . "-" . $_FILES["file"]["name"];
  }
 }

 //*** Create Scan Entry ***
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
 $result=$mysqli->query("select id from scans order by id
desc limit 1");
 $row = $result->fetch_assoc();
 $scanId=$row['id'];

 //*** Create Records Entries ***
 $xmlfile=$file;
```

```
 $fp = fopen($xmlfile, 'r');
 $xmldata = fread($fp,filesize($xmlfile));
 fclose($fp);
 $p = xml_parser_create();
 xml_parser_set_option($p,XML_OPTION_SKIP_WHITE,1);
 xml_parse_into_struct($p, $xmldata, $vals, $index);
 xml_parser_free($p);
 $mVars=array('tag','attributes');
 $groupStarted=0;
 foreach($vals as $key=>$val){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  foreach($val as $key2=>$val2){
   $$key2=$val2;
  }
  if((isset($tag))&&($tag=="CDF:SELECT")){
   continue;
  }
  foreach($mVars as $mVar){
   if(($mVar=="attributes")&&(is_array($attributes))){
    foreach($attributes as $aKey=>$aVal){
     $$aKey=$aVal;
    }
   }
  }
  if(($level==3)&&($type=="open")&&($tag=="CDF:RULE-
RESULT")){
   $ruleId=$IDREF;
  }

if(($level==4)&&($type=="complete")&&($tag=="CDF:RESULT")){
   $result=$value;
  }

if(($level==4)&&($type=="complete")&&($tag=="CDF:IDENT")){
   $identCci=$value;
  }
  if(($ruleId)&&($result)&&($tag=="CDF:RULE-
RESULT")&&($level==3)&&($type=="close")){
   $date=preg_split("[T]",$TIME);
   $time=$date[1];
```

```
   $date=$date[0];
   $year=preg_split("[-]",$date);
   $day=$year[2];
   $month=$year[1];
   $year=$year[0];
   $time=str_replace("-","",$time);
   $timestamp = strtotime("$year-$month-$day $time");
   $sql="insert into results
(ruleId,result,identCci,timestamp,scanId) values
('$ruleId','$result','$identCci','$timestamp','$scanId');";
   print "$sql<br>";
   $mysqli->query($sql);
   $lastTimestamp=$timestamp;
   $ruleId='';
   $result='';
   $identCci='';
   $time='';
   $timestamp='';
   $year='';
   $month='';
   $day='';
   $date='';
   $sql='';
  }
 }
 $sql="update scans set timestamp=$lastTimestamp where
id=$scanId";
 $mysqli->query($sql);
}

//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){
  $$var='';
 }
}

//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
```

```
 $result->close();
}

//*** Form Header ***
print "<form enctype='multipart/form-data' action=$phpSelf
method=post style=margin-bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";

//*** Form ***
print "

<tr><td class=formtag>Host:</td><td class=formfield>
<select name=hostId>
<option value=''>--SELECT--</option>
";
$sql="select id,name from hosts";
$result = $mysqli->query($sql);
while ($row = $result->fetch_assoc()){
 $hostId=$row['id'];
 $hostName=$row['name'];
 print "<option value=$hostId>$hostName</option>";
}
print "
</select></td></tr>
<tr><td class=formtag>Platform:</td><td
class=formfield><input type=text size=40
name=platform></td></tr>
<tr><td class=formtag>File:</td><td class=formfield><input
type=file name=file></td></tr>



";

//*** Form Footer ***
if($mode=="none"){
 $mode="add";
}
if($mode=="edit"){
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
```

```
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";

//*** BROWSE ***
print "<table class=list>";
print "<tr>";
$vars=array('id','name','timestamp','file');
foreach($vars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "</tr>";
$sql="select * from $dbTable";
$sql="select s.id id, s.timestamp timestamp, s.file file,
h.name name from scans s join hosts h on s.hostId=h.id";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
   if($var=="timestamp"){
    $timestamp=date("m/d/y",$timestamp);
   }
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc><a
href=$phpSelf?mode=edit&id=$id>${$var}</a></td>";
    }else{
```

```
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
?>
```

O. UPLOADCONFIG.PHP

```
<?php
include "includes.php";
$now=time();
if(isset($_POST['scanmode'])){

 //*** Create Scan Entry ***
 print "<table class=list>";
 $scriptDir="scanbox";
 $sessionConfigId=$_SESSION['configId'];

 //*** Copy Config File ***
 $sql="select hostId,file from configs where
id=$sessionConfigId";
 $res = $mysqli->query($sql);
 $row = $res->fetch_assoc();
 $configFile=$row['file'];
 $hostId=$row['hostId'];
 $mysqli->query("insert into scans (hostId,timestamp)
values ($hostId,$now)");
 $res=$mysqli->query("select id from scans order by id desc
limit 1");
 $row = $res->fetch_assoc();
 $scanId=$row['id'];
 copy("$configFile","$scriptDir/device.cfg");

 //*** Create Check Files ***
 $sql="select g.version version,g.identCci
identCci,g.ruleId ruleId, g.title title,c.code code from
profilesMap pm join profiles p on (p.id = pm.profileId)
join groups g on (pm.vulnId = g.vulnId) left join code c on
```

```
(c.groupId=g.id) where p.documentId='$documentId' and
p.id='$profileId' and c.code is not null";
 $res = $mysqli->query($sql);
 while($row = $res->fetch_assoc()){
  $version=$row['version'];
  $code=$row['code'];
  $identCci=$row['identCci'];
  $ruleId=$row['ruleId'];
  $title=$row['title'];
  if(file_exists($version)){unlink($version);}
  $thisFile="$scriptDir/$version.sh";
  $thisHandle = fopen($thisFile, 'w') or die("can't open
file");
  fwrite($thisHandle, $code);
  fclose($thisHandle);
  `dos2unix $thisFile`;
  chmod($thisFile,0777);
  chdir($scriptDir);
  $output=`./$version.sh`;
  $status=substr($output,0,1);
  if($status=="0"){$result="pass";}else{$result="fail";}
  $output=substr($output,1);
  $sql2="insert into results
(timestamp,ruleId,result,identCci,scanId,output,status)
values
($now,'$ruleId','$result','$identCci',$scanId','$output','$
status')";
  $mysqli->query($sql2);
  chdir("../");
  if($cc==1){$tc=$rc1;$cc=0;}else{$tc=$rc2;$cc=1;}
  print "<tr><td class=list bgcolor=$tc>$version $identCci
-$status-</td><td class=list
bgcolor=$tc><pre>$output</pre></td></tr>";
 }
 chdir("../../");
 print "</table>";
}

$dbTable="configs";
$vars=getFields($dbTable);
$now=time();
$timestamp=time();

//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){$$var=$_POST["$var"];}
```

```
 if(isset($_GET["$var"])){$$var=$_GET["$var"];}
}

//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
}

//*** Add Record ***
if($mode=="add"){
 //*** Upload File ***
 $temp = explode(".", $_FILES["file"]["name"]);
 if ($_FILES["file"]["size"] < 2000000){
  if (!$_FILES["file"]["error"] > 0){
   move_uploaded_file($_FILES["file"]["tmp_name"],
   "uploads/" . $now . "-" . $_FILES["file"]["name"]);
   $file="uploads/" . $now . "-" . $_FILES["file"]["name"];
  }
 }

 //*** Add Scan Entry to DB ***
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";} //insert commas as needed
  if($var=="timestamp"){$$var==time();}
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}

//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){$$var='';}
}
```

```
//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
}

//*** Form Header ***
print "<form enctype='multipart/form-data' action=$phpSelf
method=post style=margin-bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";

//*** Form ***
  print "

<tr><td class=formtag>Host:</td><td class=formfield>
<select name=hostId>
<option value=''>--SELECT--</option>
";
$sql="select id,name from hosts";
$result = $mysqli->query($sql);
while ($row = $result->fetch_assoc()){
 $hostId=$row['id'];
 $hostName=$row['name'];
 print "<option value=$hostId>$hostName</option>";
}
print "
</select></td></tr>
<tr><td class=formtag>Description:</td><td
class=formfield><input type=input name=description
value='$description' size=50 ></td></tr>
<tr><td class=formtag>File:</td><td class=formfield><input
type=file name=file></td></tr>
";

//*** Form Footer ***
if($mode=="none"){$mode="add";}
if($mode=="edit"){
```

```
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";

//*** BROWSE ***
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if($var!="id") {
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "<td></td></tr>";
$sql="select * from $dbTable";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()) {
  if($cc==1){$tc=$rc1;$cc=0;}else{$tc=$rc2;$cc=1;}
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }else{
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "</tr>";
 }
 $result->close();
```

```
}
$mysqli->close();
print "</table>";
?>
```

P. SCANS.PHP

```
<?php
include "includes.php";
$dbTable="scans";

$vars=getFields($dbTable);
//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){$$var=$_POST["$var"];}
 if(isset($_GET["$var"])){$$var=$_GET["$var"];}
}
//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
}
//*** Add ***
if($mode=="add"){
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}
//*** Update Database ***
if($mode=="update"){
 $sql="update $dbTable set ";
 $count=1;
```

```
 foreach($vars as $var){
  if($count>=2){$sql.=",";}
  $sql.="$var=\"${$var}\"";
  $count++;
 }
 $sql.=" where id=$id";
 $mysqli->query($sql);
 $mode="none";
}
//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){$$var='';}}
//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
}
//*** Form Header ***
print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";
//*** Form ***
foreach($vars as $var){
 $uc_var=ucfirst($var);
 if($var!="id") {
  print "<tr><td class=formtag>$uc_var:</td><td
class=formfield><input type=text size=20 name=$var
value='${$var}'></td></tr>";
 }
}
//*** Form Footer ***
if($mode=="none"){$mode="add";}
if($mode=="edit"){
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
```

```
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";
//*** BROWSE ***
print "<table class=list>";
print "<tr>";
print "<td class=listheader>Date</td>";
print "<td class=listheader>Host</td>";
print "<td class=listheader>Platform</td>";
print "<td class=listheader>File</td>";
print "</tr>";
$sql="select s.id id, s.hostId hostId, s.timestamp
timestamp, s.file file, s.platform platform,h.name from
scans s join hosts h on s.hostId=h.id order by timestamp";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  $id=$row['id'];
  $hostId=$row['hostId'];
  $timestamp=$row['timestamp'];
  $file=$row['file'];
  $name=$row['name'];
  $platform=$row['platform'];
  $timeFormatted=date("d-M-y",$timestamp);
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  print "<tr>";
  print "<td class=list bgcolor=$tc><a
href=$phpSelf?mode=edit&id=$id>$timeFormatted</a></td>";
  print "<td class=list bgcolor=$tc>$name</td>";
  print "<td class=list bgcolor=$tc>$platform</td>";
  print "<td class=list bgcolor=$tc>$file</td>";
  print "</tr>";
 }
 $result->close();
```

```
}
$mysqli->close();
print "</table>";
?>
```

Q. CONFIGS.PHP

```
<?php
include "includes.php";
$dbTable="configs";
if(isset($_POST['configId'])){
 print "Config has been selected<br>";
 $_SESSION['configId']=$_POST['configId'];
}
$vars=getFields($dbTable);

//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){
  $$var=$_POST["$var"];
 }
 if(isset($_GET["$var"])){
  $$var=$_GET["$var"];
 }
}

//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 $mode="none";
}

//*** Add ***
if($mode=="add"){
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
```

```
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}

//*** Update Database ***
if($mode=="update"){
 $sql="update $dbTable set ";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.="$var=\"${$var}\"";
  $count++;
 }
 $sql.=" where id=$id";
 $mysqli->query($sql);
 $mode="none";
}

//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){
  $$var='';
 }
}

//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $result = $mysqli->query($sql);
 while ($row = $result->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $result->close();
}
$configText=file_get_contents($file);
```

```
//*** Form Header ***
print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";

//*** Form ***
foreach($vars as $var){
 $uc_var=ucfirst($var);
 if($var!="id"){
  print "<tr><td class=formtag>$uc_var:</td><td
class=formfield><input type=text size=20 name=$var
value='${$var}'></td></tr>";
 }
}
print "<tr><td class=formtag>Text:</td><td class=formfield
style='color: #F0F0F0; background-color: #181818;'>";
print "<pre>";
print "$configText";
print "</pre>";
print "</td></tr>";

//*** Form Footer ***
if($mode=="none"){
 $mode="add";
}
if($mode=="edit"){
 print "<input type=hidden name=id value=$id>";
 $mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";

//*** BROWSE ***
```

```
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "</tr>";

$sql="select * from $dbTable";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc><a
href=$phpSelf?mode=edit&id=$id>${$var}</a></td>";
    }else{
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "<td class=list bgcolor=$tc><form action=$phpSelf
method=post style=margin-bottom:0;><input type=hidden
name=configId value=$id><input type=submit
value=select></form></td>";
  print "</tr>";
 }
 $result->close();
}
$mysqli->close();
print "</table>";
?>
```

R. REVIEWSCANS.PHP

```
<?php
include "includes.php";
$dbTable="scans";
$vars=getFields($dbTable);
$now=time();
$timestamp=time();

//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){
  $$var=$_POST["$var"];
 }
 if(isset($_GET["$var"])){
  $$var=$_GET["$var"];
 }
}

//*** BROWSE ***
print "<table class=list>";
print "<tr>";
print "<td class=listheader>Date</td>";
print "<td class=listheader>Host</td>";
print "<td class=listheader>Platform</td>";
print "<td class=listheader>Baseline</td>";
print "<td class=listheader>Target</td>";
print "</tr>";
print "<form action=$phpSelf method=post>";
$sql="select s.platform platform,s.id id,s.timestamp
timestamp, h.name host from scans s join hosts h on
s.hostId=h.id order by timestamp";
if ($result = $mysqli->query($sql)){
 while ($row = $result->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  $timestamp=$row["timestamp"];
  $id=$row["id"];
  $host=$row["host"];
  $platform=$row["platform"];
  $dateFormatted=date("m-d-Y g:ma",$timestamp);
```

```
  print "<tr>";
  print "<td class=list bgcolor=$tc>$dateFormatted</td>";
  print "<td class=list bgcolor=$tc>$host</td>";
  print "<td class=list bgcolor=$tc>$platform</td>";
  print "<td class=list bgcolor=$tc><input type=radio
name=baseline value=$id></td>";
  print "<td class=list bgcolor=$tc><input type=radio
name=target value=$id></td>";
  print "</tr>";
 }
 $result->close();
}
print "<tr><td colspan=30 align=center><input type=submit
value=submit></td></tr>";
print "</table></form>";
if(isset($_POST['baseline'])){
 $baselineId=$_POST['baseline'];
}else{
 $baselineId='';
}
if(isset($_POST['target'])){
 $otherId=$_POST['target'];
}else{
 $otherId='';
}
if($baselineId){
 print "<table class=list>";
 print "<td class=listheader>Vuln ID</td>";
 print "<td class=listheader>Ident CCI</td>";
 print "<td class=listheader>Rule ID</td>";
 print "<td class=listheader>Rule</td>";
 print "<td class=listheader>Baseline</td>";
 print "<td class=listheader>Target</td>";
 print "<td class=listheader></td>";
 $sql="select g.vulnId gVulnId, b.identCci
bIdentCci,g.title title, b.id bResultId, b.ruleId bRuleId,
b.result bResult from results b join groups g on
b.ruleId=g.ruleId where scanId=$baselineId";
 if ($res = $mysqli->query($sql)){
  while ($row = $res->fetch_assoc()){
   if($cc==1){
    $tc=$rc1;
    $cc=0;
   }else{
    $tc=$rc2;
    $cc=1;
```

```
   }
   $title=$row['title'];
   $identCci=$row['bIdentCci'];
   $vulnId=$row['gVulnId'];
   $ruleId=$row['bRuleId'];
   $bResultId=$row['bResultId'];
   $bResult=$row['bResult'];
   $sql2="select result from results where ruleId='$ruleId'
and scanId=$otherId";
   $res2 = $mysqli->query($sql2);
   $row2 = $res2->fetch_assoc();
   $cResult=$row2['result'];
   print "<tr>";
   if($bResult!=$cResult){
    $tc="#FF6666";
   }else{
    $tc="#c0c0c0";
   }
   print "<td class=list bgcolor=$tc>$vulnId</td>";
   print "<td class=list bgcolor=$tc>$identCci</td>";
   print "<td class=list bgcolor=$tc>$ruleId</td>";
   print "<td class=list bgcolor=$tc>$title</td>";
   print "<td class=list bgcolor=$tc>$bResult</td>";
   print "<td class=list bgcolor=$tc>$cResult</td>";
   print "<td class=list bgcolor=$tc>";
   print "<form action=results.php method=post
style=margin-bottom:0;>";
   print "<input type=hidden name=id value=$bResultId>";
   print "<input type=submit value=edit>";
   print "<input type=hidden name=mode value=edit>";
   print "</form>";
   print "</td>";
   print "</tr>";
  }
  print "</table>";
 }
}
?>
```

S. RESULTS.PHP

```
<?php
include "includes.php";
$dbTable="results";
$vars=getFields($dbTable);
```

```
//*** Get Variables ***
foreach($vars as $var){
 if(isset($_POST["$var"])){
  $$var=$_POST["$var"];
 }
 if(isset($_GET["$var"])){
  $$var=$_GET["$var"];
 }
}

//*** Delete Record ***
if($mode=="delete"){
 $sql="delete from $dbTable where id=$id";
 $res = $mysqli->query($sql);
 $mode="none";
}

//*** Add ***
if($mode=="add"){
 $sql="insert into $dbTable (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.=$var;
  $count++;
 }
 $sql.=") values (";
 $count=1;
 foreach($vars as $var){
  if($count>=2){
   $sql.=",";
  }
  $sql.="\"${$var}\"";
  $count++;
 }
 $sql.=")";
 $mysqli->query($sql);
}

//*** Update Database ***
if($mode=="update"){
 $sql="update $dbTable set ";
 $count=1;
 foreach($vars as $var){
```

```
  if($count>=2){
   $sql.=",";
  }
  $sql.="$var=\"${$var}\"";
  $count++;
 }
 $sql.=" where id=$id";
 $mysqli->query($sql);
 $mode="none";
}

//*** Define Variables ***
if(($mode=="add")||($mode=="none")){
 foreach($vars as $var){
  $$var='';
 }
}

//*** Query DB for Edit ***
if($mode=="edit"){
 $sql="select * from $dbTable where id=$id";
 $res = $mysqli->query($sql);
 while ($row = $res->fetch_assoc()){
  foreach($vars as $var){
   $$var=$row["$var"];
  }
 }
 $res->close();
}

//*** Form Header ***
print "<form action=$phpSelf method=post style=margin-
bottom:0;><table class=form>";
$uc_page=ucfirst($page);
print "<tr><td colspan=2
class=formTitle>$uc_page</td></tr>";

//*** Form ***
foreach($vars as $var){
 $uc_var=ucfirst($var);
 if($var!="id"){
  print "<tr><td class=formtag>$uc_var:</td><td
class=formfield><input type=text size=20 name=$var
value='${$var}'></td></tr>";
 }
}
```

```
//*** Form Footer ***
if($mode=="none"){
 $mode="add";
}
if($mode=="edit"){
 print "<input type=hidden name=id
value=$id>";$mode="update";
}
print "<tr><td align=center colspan=2
class=formfooter><table align=center><tr>";
print "<td><input type=hidden name=mode value=$mode><input
type=submit value=$mode></form></td>";
if($mode=="update"){
 print "
 <form action=$phpSelf method=post style=margin-bottom:0;>
 <input type=hidden name=id value=$id>
 <input type=hidden name=mode value=delete>
 <td><input type=submit value=delete></td></form>";
}
print "</tr></table></td></tr></table>";

//*** BROWSE ***
print "<table class=list>";
print "<tr>";
foreach($vars as $var){
 if($var!="id"){
  $uc_var=ucfirst($var);
  print "<td class=listheader>$uc_var</td>";
 }
}
print "</tr>";
$sql="select * from $dbTable";
if ($res = $mysqli->query($sql)){
 while ($row = $res->fetch_assoc()){
  if($cc==1){
   $tc=$rc1;
   $cc=0;
  }else{
   $tc=$rc2;
   $cc=1;
  }
  print "<tr>";
  $col=1;
  foreach($vars as $var){
   $$var=$row["$var"];
```

```
   if($var!="id"){
    if($col==1){
     print "<td class=list bgcolor=$tc><a
href=$phpSelf?mode=edit&id=$id>${$var}</a></td>";
    }else{
     print "<td class=list bgcolor=$tc>${$var}</td>";
    }
    $col++;
   }
  }
  print "</tr>";
 }
 $res->close();
}
$mysqli->close();
print "</table>";
?>
```

LIST OF REFERENCES

[1] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 82-84, April. 1965.

[2] Hewlett-Packard Development Company, L.P. (1968). "History of the 9100A desktop calculator." [Online]. Available: http://www.hp.com/hpinfo/abouthp/histnfacts/museum/personalsystems/0021/0021history.html

[3] U.S. Bureau of Labor Statistics. CPI inflation calculator. [Online]. Available: http://data.bls.gov/cgi-bin/cpicalc.pl

[4] Raspberry Pi Foundation. Raspberry Pi FAQs. [Online]. Available: http://www.raspberrypi.org/faqs

[5] R. Meulen and C. Pettey. (2008, June). Gartner says more than 1 billion PCs in use worldwide and headed to 2 billion units by 2014. Gartner, Inc., Stamford, CT. [Online news release]. Available: http://www.gartner.com/newsroom/id/703807

[6] D. D'Agostino and G. Wilshusen. (2011, July). *Defense Department Cyber Efforts: DOD faces challenges in its cyber activities* (GAO-11-75). U.S. Government Accountability Office, Washington, DC. [Online]. Available: http://www.gao.gov/assets/330/321818.pdf

[7] Symantec Corporation. Vulnerability trends. Symantec Corporation, Mountain View, CA. [Online]. Available: http://www.symantec.com/threatreport/topic.jsp?id=vulnerability_trends&aid=total_number_of_vulnerabilities

[8] SecureState, LLC. DIACAP / DoD 8500. [Online]. Available: http://www.securestate.com/Federal/Certification%20and%20%20Accreditation/Pages/DIACAP-D0D8500.aspx

[9] "War in the fifth domain." (2010, July). *The Economist*. [Online]. Available: http://www.economist.com/node/16478792

[10] National Institute of Standards and Technology. CVE and CVE vulnerability database advanced search. [Online]. Available: http://web.nvd.nist.gov/view/vuln/search-advanced

[11] "25 Years of vulnerabilities: Linux has the most." (2013, March). *iTWire.* [Online]. Available: http://www.eitr.com.au/news/25-Years-of-vulnerabilities-Linux-has-the-most.php

[12] Computer Security Division Information Technology Laboratory. (2005, October). Advising users on information technology, *Information Technology Laboratory (ITL) Bulletin*. National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistbul/b-Oct-05.pdf

[13] Department of Defense. (2010, February). Quadrennial defense review report. Department of Defense, Washington, DC. [Online]. Available: http://www.defense.gov/qdr/images/QDR as of 12Feb10 1000.pdf

[14] National Computer Security Center. (1994, January). Introduction to certification and accreditation (NCSC-TG-029). National Computer Security Center, Fort Meade, MD. [Online]. Available: http://csrc.nist.gov/publications/secpubs/otherpubs/CA Handbook.pdf

[15] Information Assurance Training Center. Lesson 11: Department of Defense Information Assurance Certification and Accreditation Process. [Online]. Available: https://ia.signal.army.mil/IAF/IASOLesson11.asp

[16] Computer Security Division Information Technology Laboratory (2010, February). Guide for applying the risk management framework to federal information systems (special publication 800-37 Rev. 1). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-37-rev1/sp800-37-rev1-final.pdf

[17] "Thoughts on software assurance." (2009, May). *Richard Bejtlich's TAOSecurity Blog*. [Online]. Available: http://taosecurity.blogspot.com/2005/09/thoughts-on-software-assurance-last.html

[18] P. Buxbaum, "Automatic for Security," *Military Information Technology*, vol. 16, no. 4, p. 7, May. 2012.

[19] S. Quinn et al. (2012, January). Guide to adopting and using the security content automation protocol (SCAP) (Ver. 1.2 NIST Special Publication 800-117 Rev. 1). [Draft]. National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/drafts/800-117-R1/Draft-SP800-117-r1.pdf

[20] The Mitre Corporation. OVAL language overview. [Online]. Available: http://oval.mitre.org/language/about/overview.html

[21] D. Waltermire et al. (2011, April). Specification for the open checklist interactive language (OCIL) Version 2.0 (Report 7692). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7692/nistir-7692.pdf

[22] N. Ziring and S. D. Quinn. (2012, March). Specification for the extensible configuration checklist description format (XCCDF) Ver. 1.2 Rev. 4 (Report 7275). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7275r3/NISTIR-7275r3.pdf

[23] D. Waltermire and K. Scarfone. (2011, February). Guide to using vulnerability naming schemes (Special Publication 800-51 Rev. 1). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-51-rev1/SP800-51rev1.pdf

[24] P. Mell et al. (2007, August). The common vulnerability scoring system (CVSS) and its applicability to federal agency systems (Report 7435). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7435/NISTIR-7435.pdf

[25] A. Halbardier et al.(2011, June). Specification for the asset reporting format 1.1 (Report 7694). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7694/NISTIR-7694.pdf

[26] H. Booth and A. Halbardier. (2011, September). Trust model for security automation data 1.0 (TMSAD) (Report 7802). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistir/ir7802/NISTIR-7802.pdf

[27] Tenable Network Security, Inc. (2012). Tenable delivers best-of-breed configuration compliance and vulnerability management for U.S. Department of Defense. Tenable Network Security, Inc., Columbia, MD. [Online]. Available: http://www.satisnet.co.uk/pdfs/tenable_acas_cs_v1_web.pdf

[28] Tenable Network Security. Tenable passive vulnerability scanner data sheet. [Online]. Available: http://www.tenable.com/sites/drupal.dmz.tenablesecurity.com/files/datasheets/PVS_DS_(EN)_v5_web.pdf

[29] User's guide and help desk/troubleshooting guide continuous monitoring and risk scoring (CMRS) Enterprise Release 1.1 (unpublished). Defense Information Systems Agency , Scott Air Force Base, IL, 2013.

[30] Computer Security Division Information Technology Laboratory. (2011, August). Guide for security-focused configuration management of information systems (Special Publication 800-128). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-128/sp800-128.pdf

[31] Computer Security Division Information Technology Laboratory. (2009, August). Recommended security controls for federal information systems and organizations (Special Publication 800-53). National Institute of Standards and Technology, Gaithersburg, MD. [Online]. Available: http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf

[32] C. Ramey. (2011, June 8). "The Bourne again shell," in *The Architecture of Open Source Applications*, K. Bostic et al. [Online]. Available: http://www.aosabook.org/en/bash.html

[33] C. Poe. (2012, September 19). *Beginning Perl*. [Online]. Available: http://it-ebooks.info/book/977/

[34] "Why I use Perl...and will continue to do so." (2013, February). *Dr Drobb's World of Software Development*. [Online]. Available: http://www.drdobbs.com/open-source/why-i-use-perland-will-continue-to-do-so/240148364

[35] C. Hopkins, *Jump Start PHP*. Collingwood, Austalia: SitePoint Pty. Ltd, 2013.

[36] Kristofer Layon, *Mobilizing Web Sites: Develop and Design*. Berkeley, CA: Peachpit Press, 2011.

[37] *Wikipedia*. List of Apache–MySQL–PHP packages. [Online]. Available: http://en.wikipedia.org/wiki/List of Apache%E2%80%93MySQL%E2%80%93PHP packages

[38] *Wikipedia*. Relational database. [Online]. Available: http://en.wikipedia.org/wiki/Relational database

[39] Sideris Corporation, *Data Modeling: Logical Database Design*. Newton, MA: Sideris Courseware Corporation, 2011.

[40] Netcraft LTD. December 2013 Web server survey. [Online]. Available: http://news.netcraft.com/archives/2013/12/06/december-2013-web-server-survey.html

[41] Microsoft. Installing IIS 7. [Online]. Available: http://www.iis.net/learn/install

[42] The Apache Software Foundation. Downloading the Apache HTTP server. [Online]. Available: http://httpd.apache.org/download.cgi

[43] The United States Navy. Security Content Automation Protocol (SCAP) compliance checker. [Online]. Available: http://www.public.navy.mil/spawar/Atlantic/ProductsServices/Pages/SCAP.aspx

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California